lab-solution

September 27, 2023

1 Assignment: Linear regression on the Advertising data

Fraida Fund

Student Name: *Yinhao Liu*
Net ID: N10239426

Submit answers to the questions in PrairieLearn as you work through this notebook.

To illustrate principles of linear regression, we are going to use some data from the textbook "An Introduction to Statistical Learning withApplications in R" (Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani) (available via NYU Library).

The dataset is described as follows:

> Suppose that we are statistical consultants hired by a client to provide advice on how to improve sales of a particular product. The `Advertising` data set consists of the sales of that product in 200 different markets, along with advertising budgets for the product in each of those markets for three different media: TV, radio, and newspaper.
>
> ...
>
> It is not possible for our client to directly increase sales of the product. On the other hand, they can control the advertising expenditure in each of the three media. Therefore, if we determine that there is an association between advertising and sales, then we can instruct our client to adjust advertising budgets, thereby indirectly increasing sales. In other words, our goal is to develop an accurate model that can be used to predict sales on the basis of the three media budgets.

Sales are reported in thousands of units, and TV, radio, and newspaper budgets, are reported in thousands of dollars.

For this assignment, you will fit a linear regression model to a small dataset. You will iteratively improve your linear regression model by examining the residuals at each stage, in order to identify problems with the model.

Make sure to include your name and net ID in a text cell at the top of the notebook.

[80]:
```
from sklearn import metrics
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

import numpy as np
```

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
sns.set()

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

1.0.1 0. Read in and pre-process data

In this section, you will read in the "Advertising" data, and make sure it is loaded correctly. Visually inspect the data using a pairplot, and note any meaningful observations. In particular, comment on which features appear to be correlated with product sales, and which features appear to be correlated with one another. Then, split the data into training data (70%) and test data (30%).

The code in this section is provided for you.

Read in data

[81]:
```
!wget 'https://www.statlearning.com/s/Advertising.csv' -O 'Advertising.csv'
```

```
--2023-09-27 06:00:00--  https://www.statlearning.com/s/Advertising.csv
Resolving www.statlearning.com (www.statlearning.com)… 198.185.159.144,
198.49.23.145, 198.185.159.145, …
Connecting to www.statlearning.com
(www.statlearning.com)|198.185.159.144|:443… connected.
HTTP request sent, awaiting response… 302 Found
Location: https://static1.squarespace.com/static/5ff2adbe3fe4fe33db902812/t/5fff
e03b4091076ff5b30c72/1610604603901/Advertising.csv [following]
--2023-09-27 06:00:01--  https://static1.squarespace.com/static/5ff2adbe3fe4fe33
db902812/t/5fffe03b4091076ff5b30c72/1610604603901/Advertising.csv
Resolving static1.squarespace.com (static1.squarespace.com)… 151.101.0.238,
151.101.64.238, 151.101.128.238, …
Connecting to static1.squarespace.com
(static1.squarespace.com)|151.101.0.238|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 4555 (4.4K) [text/csv]
Saving to: 'Advertising.csv'

Advertising.csv     100%[===================>]   4.45K  --.-KB/s    in 0s

2023-09-27 06:00:01 (66.0 MB/s) - 'Advertising.csv' saved [4555/4555]
```

[82]:
```
df  = pd.read_csv('Advertising.csv', index_col=0)
df.head()
```

[82]:

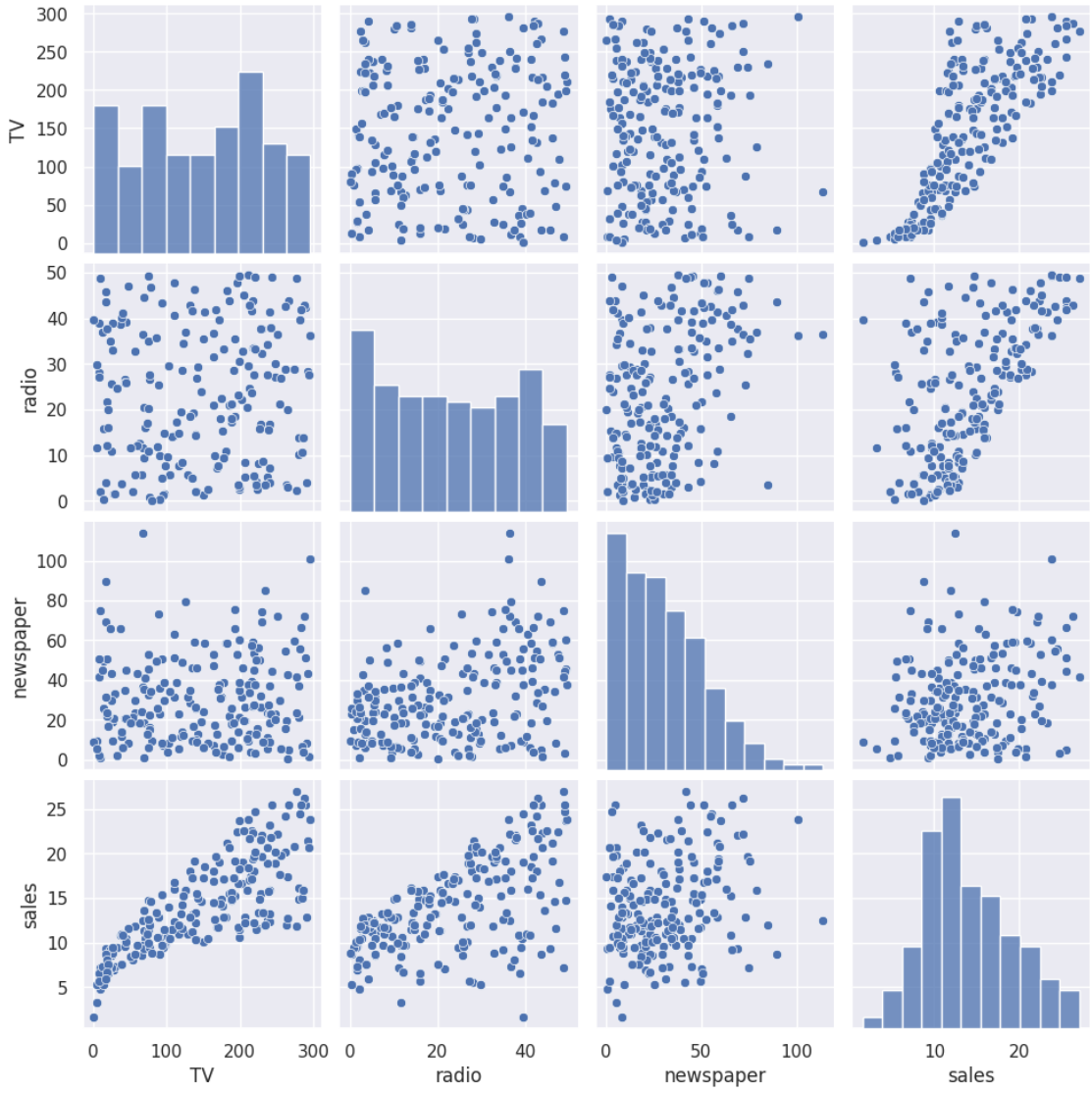
	TV	radio	newspaper	sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

Note that in this dataset, the first column in the data file is the row label; that's why we use `index_col=0` in the `read_csv` command. If we would omit that argument, then we would have an additional (unnamed) column in the dataset, containing the row number.

(You can try removing the `index_col` argument and re-running the cell above, to see the effect and to understand why we used this argument.)

Visually inspect the data

[83]:
```
sns.pairplot(df);
```



The most important panels here are on the bottom row, where `sales` is on the vertical axis and the advertising budgets are on the horizontal axes.

Looking at this row, it appears that TV ad spending and radio ad spending are likely to be useful predictive features for `sales`; for newspaper ad spending, it is not clear from the pairplot whether there is a relationship.

Split up data We will use 70% of the data for training and the remaining 30% to evaluate the regression model on data *not* used for training.

[84]:
```
train, test = train_test_split(df, test_size=0.3, random_state=9)
```

We will set the `random_state` to a constant so that every time you run this notebook, exactly the same data points will be assigned to test vs. training sets. This is helpful in the debugging stage.

[85]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 140 entries, 134 to 127
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   TV         140 non-null    float64
 1   radio      140 non-null    float64
 2   newspaper  140 non-null    float64
 3   sales      140 non-null    float64
dtypes: float64(4)
memory usage: 5.5 KB
```

[86]:
```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 60 entries, 85 to 7
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   TV         60 non-null     float64
 1   radio      60 non-null     float64
 2   newspaper  60 non-null     float64
 3   sales      60 non-null     float64
dtypes: float64(4)
memory usage: 2.3 KB
```

1.0.2 1. Fit simple linear regression models

Use the training data to fit a simple linear regression to predict product sales, for each of three features: TV ad budget, radio ad budget, and newspaper ad budget. In other words, you will

fit *three* regression models, with each model being trained on one feature. For each of the three regression models, create a plot of the training data and the regression line, with product sales (y) on the vertical axis and the feature on which the model was trained (x) on the horizontal axis.

Also, for each regression model, print the intercept and coefficients, and compute the MSE and R2 on the training data, and MSE and R2 on the test data.

Comment on the results. Which type of ad spending seems to be associated with the largest increase in product sales? Which regression model is most effective at predicting product sales?

The code in this section is provided for you. However, you will need to add comments, observations, and answers to the questions.

Fit a simple linear regression

[87]:
```
reg_tv    = LinearRegression().fit(train[['TV']], train['sales'])
reg_radio = LinearRegression().fit(train[['radio']], train['sales'])
reg_news  = LinearRegression().fit(train[['newspaper']], train['sales'])
```

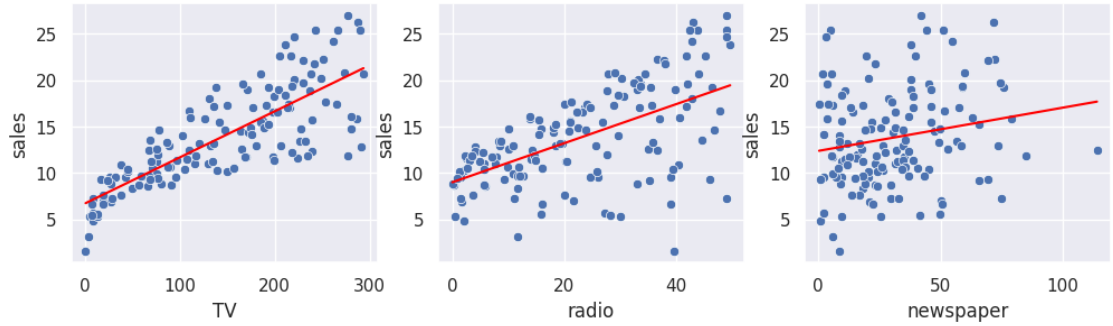
Look at coefficients

[88]:
```
print("TV       : ", reg_tv.coef_[0], reg_tv.intercept_)
print("Radio    : ", reg_radio.coef_[0], reg_radio.intercept_)
print("Newspaper: ", reg_news.coef_[0], reg_news.intercept_)
```

```
TV       :  0.04964468781898984 6.711432632336138
Radio    :  0.21062312839115208 8.997640913704718
Newspaper:  0.046574464282301664 12.375549417451523
```

Plot data and regression line

[89]:
```
fig = plt.figure(figsize=(12,3))

plt.subplot(1,3,1)
sns.scatterplot(data=train, x="TV", y="sales");
sns.lineplot(data=train, x="TV", y=reg_tv.predict(train[['TV']]), color='red');

plt.subplot(1,3,2)
sns.scatterplot(data=train, x="radio", y="sales");
sns.lineplot(data=train, x="radio", y=reg_radio.predict(train[['radio']]),␣
 ↪color='red');

plt.subplot(1,3,3)
sns.scatterplot(data=train, x="newspaper", y="sales");
sns.lineplot(data=train, x="newspaper", y=reg_news.
 ↪predict(train[['newspaper']]), color='red');
```



Compute R2, MSE for simple regression

[90]:
```
y_pred_tr_tv    = reg_tv.predict(train[['TV']])
y_pred_tr_radio = reg_radio.predict(train[['radio']])
y_pred_tr_news  = reg_news.predict(train[['newspaper']])
```

[91]:
```
r2_tr_tv    = metrics.r2_score(train['sales'], y_pred_tr_tv)
r2_tr_radio = metrics.r2_score(train['sales'], y_pred_tr_radio)
r2_tr_news  = metrics.r2_score(train['sales'], y_pred_tr_news)
print("TV       : ", r2_tr_tv)
print("Radio    : ", r2_tr_radio)
print("Newspaper: ", r2_tr_news)
```

```
TV       :  0.6462575775839753
Radio    :  0.33630082549935214
Newspaper:  0.0373981756207491
```

[92]:
```
mse_tr_tv    = metrics.mean_squared_error(train['sales'], y_pred_tr_tv)
mse_tr_radio = metrics.mean_squared_error(train['sales'], y_pred_tr_radio)
mse_tr_news  = metrics.mean_squared_error(train['sales'], y_pred_tr_news)
print("TV       : ", mse_tr_tv)
print("Radio    : ", mse_tr_radio)
print("Newspaper: ", mse_tr_news)
```

```
TV       :  9.798510609335318
Radio    :  18.384177273212142
Newspaper:  26.663650133692155
```

[93]:
```
y_pred_ts_tv    = reg_tv.predict(test[['TV']])
y_pred_ts_radio = reg_radio.predict(test[['radio']])
y_pred_ts_news  = reg_news.predict(test[['newspaper']])
```

[94]:
```
r2_ts_tv    = metrics.r2_score(test['sales'], y_pred_ts_tv)
r2_ts_radio = metrics.r2_score(test['sales'], y_pred_ts_radio)
r2_ts_news  = metrics.r2_score(test['sales'], y_pred_ts_news)
```

```
print("TV       : ", r2_ts_tv)
print("Radio    : ", r2_ts_radio)
print("Newspaper: ", r2_ts_news)
```

```
TV       :  0.5138892470208256
Radio    :  0.3072356147167632
Newspaper:  0.06497948830922318
```

[95]:
```
mse_ts_tv    = metrics.mean_squared_error(test['sales'], y_pred_ts_tv)
mse_ts_radio = metrics.mean_squared_error(test['sales'], y_pred_ts_radio)
mse_ts_news  = metrics.mean_squared_error(test['sales'], y_pred_ts_news)
print("TV       : ", mse_ts_tv)
print("Radio    : ", mse_ts_radio)
print("Newspaper: ", mse_ts_news)
```

```
TV       :  12.288041294264643
Radio    :  17.511888641395615
Newspaper:  23.635705625160178
```

1.0.3 Comment

1. Radio advertising expenditure exhibits a positive linear relationship with sales and has the highest regression coefficient, 0.2106, which means radio advertising spending is highly associated with the largest increase in product sales.
2. Since TV model has the biggest R2 (0.6462) and the least MSE (9.798), the TV regression model is most effective at predicting product sales.

1.0.4 2. Explore the residuals for the single linear regression models

We know that computing MSE or R2 is not sufficient to diagnose a problem with a linear regression.

Create some additional plots as described below to help you identify any problems with the regression. Use training data for all of the items below.

For each of the three regression models, you will compute the residuals $(y - \hat{y})$. Then, you'll create three plots - each with three subplots, one for each regression model - as follows:

Plot 1: Create a scatter plot of predicted sales $(\hat{y})$ on the vertical axis, and actual sales (y) on the horizontal axis. Make sure both axes use the same scale (the range of the vertical axis should be the same as the range of the horizontal axis) *and* that all three subplots use the same scale. Label each axes, and each plot. What would you expect this plot to look like for a model that explains the data well?

Plot 2: Create a scatter plot with the residuals $(y - \hat{y})$ on the vertical axis, and actual sales (y) on the horizontal axis. Use the same vertical scale for all three subplots, and the same horizontal scale for all three subplots (but the vertical scale and the horizontal scale will not be the same as one another!). Comment on your observations. Is there a pattern in the residuals (and if so, what might it indicate), or do they appear to have no pattern with respect to actual sales?

Plot 3: For each of the three regression models AND each of the three features, create a scatter plot with the residuals $(y - \hat{y})$ on the vertical axis, and the feature (x) on the horizontal axis.

This plot will include nine subplots in total, for every combination of regression model and feature. Use the same vertical scale for all subplots (but the horizontal scale will depend on the feature!) Make sure to clearly label each axis, and also label each subplot with a title that indicates which regression model it uses. Is there a pattern in the residuals (and if so, what might it indicate), or do they appear to have no pattern with respect to each of the three features?

The code in this section is not provided for you. You will need to write code, as well as comments, observations, and answers to the questions.

Note that in general, to earn full credit, plots must:

- Be readable (especially text size).
- Have a label on each axis.
- Have an appropriate range for each axis. When there are multiple subplots, if the goal is to compare similar things in different subplots, in most cases it is appropriate for them all to use the same range.
- If there are multiple subplots, or multiple data series in the same plot, it must be made clear which is which.

Plot 1: Create a scatter plot of predicted sales ($\hat{y}$) on the vertical axis, and actual sales (y) on the horizontal axis. Make sure both axes use the same scale (the range of the vertical axis should be the same as the range of the horizontal axis) *and* that all three subplots use the same scale. Label each axes, and each plot. What would you expect this plot to look like for a model that explains the data well?

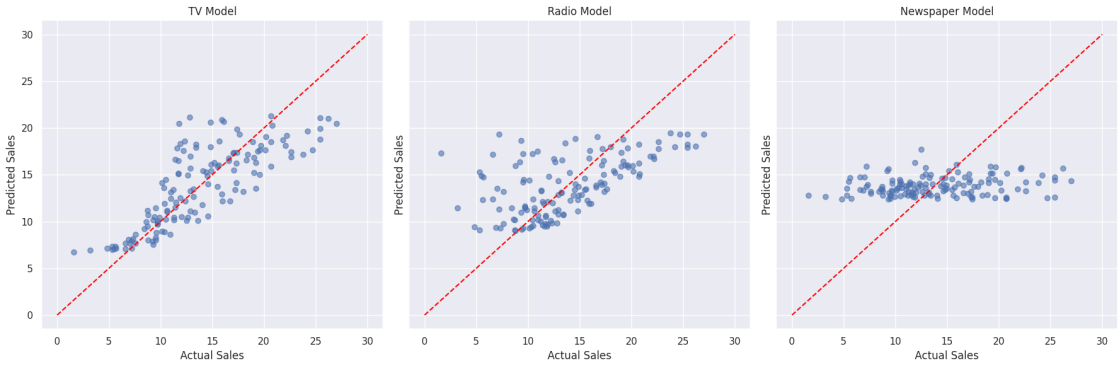
1.0.5 Comment

I hope that all the points in the graph are as close as possible to the red dashed line y = x.

[96]:
```
# Calculating predicted sales for each regression model again
train['pred_sales_tv'] = reg_tv.predict(train[['TV']])
train['pred_sales_radio'] = reg_radio.predict(train[['radio']])
train['pred_sales_news'] = reg_news.predict(train[['newspaper']])

# Plotting again
fig, axes = plt.subplots(1, 3, figsize=(18, 6), sharex=True, sharey=True)

# TV Model
axes[0].scatter(train['sales'], train['pred_sales_tv'], alpha=0.6)
axes[0].plot([0, 30], [0, 30], '--', color='red')
axes[0].set_title('TV Model')
axes[0].set_xlabel('Actual Sales')
axes[0].set_ylabel('Predicted Sales')

# Radio Model
axes[1].scatter(train['sales'], train['pred_sales_radio'], alpha=0.6)
axes[1].plot([0, 30], [0, 30], '--', color='red')
axes[1].set_title('Radio Model')
```

```
axes[1].set_xlabel('Actual Sales')
axes[1].set_ylabel('Predicted Sales')

# Newspaper Model
axes[2].scatter(train['sales'], train['pred_sales_news'], alpha=0.6)
axes[2].plot([0, 30], [0, 30], '--', color='red')
axes[2].set_title('Newspaper Model')
axes[2].set_xlabel('Actual Sales')
axes[2].set_ylabel('Predicted Sales')

plt.tight_layout()
plt.show()
```

```
[96]: <matplotlib.collections.PathCollection at 0x7a1614c75660>

[96]: [<matplotlib.lines.Line2D at 0x7a16145711b0>]

[96]: Text(0.5, 1.0, 'TV Model')

[96]: Text(0.5, 0, 'Actual Sales')

[96]: Text(0, 0.5, 'Predicted Sales')

[96]: <matplotlib.collections.PathCollection at 0x7a1614573310>

[96]: [<matplotlib.lines.Line2D at 0x7a16145739d0>]

[96]: Text(0.5, 1.0, 'Radio Model')

[96]: Text(0.5, 0, 'Actual Sales')

[96]: Text(0, 0.5, 'Predicted Sales')

[96]: <matplotlib.collections.PathCollection at 0x7a16145736a0>

[96]: [<matplotlib.lines.Line2D at 0x7a16145700a0>]

[96]: Text(0.5, 1.0, 'Newspaper Model')

[96]: Text(0.5, 0, 'Actual Sales')

[96]: Text(0, 0.5, 'Predicted Sales')
```



[97]:
```
train
```

[97]:
```
        TV  radio  newspaper  sales  pred_sales_tv  pred_sales_radio  \
134  219.8   33.5       45.1   19.6      17.623335         16.053516
137   25.6   39.0        9.3    9.5       7.982337         17.211943
155  187.8   21.1        9.5   15.6      16.034705         13.441789
178  170.2    7.8       35.2   11.7      15.160958         10.640501
69   237.4   27.5       11.0   18.9      18.497082         14.789777
..     …     …          …      …            …                …
57     7.3   28.1       41.4    5.5       7.073839         14.916151
183   56.2    5.7       29.7    8.7       9.501464         10.198193
200  232.1    8.6        8.7   13.4      18.233965         10.809000
93   217.7   33.5       59.0   19.4      17.519081         16.053516
127    7.8   38.9       50.6    6.6       7.098661         17.190881

     pred_sales_news
134        14.476058
137        12.808692
155        12.818007
178        14.014971
69         12.887869
..               …
57         14.303732
183        13.758811
200        12.780747
93         15.123443
127        14.732217

[140 rows x 7 columns]
```

Plot 2: Create a scatter plot with the residuals ($y - \hat{y}$) on the vertical axis, and actual sales (y) on the horizontal axis. Use the same vertical scale for all three subplots, and the same horizontal scale for all three subplots (but the vertical scale and the horizontal scale will not be the same as one another!). Comment on your observations. Is there a pattern in the residuals (and if so, what

might it indicate), or do they appear to have no pattern with respect to actual sales?

[98]:
```
# Calculating residuals for each regression model
train['residual_tv'] = train['sales'] - train['pred_sales_tv']
train['residual_radio'] = train['sales'] - train['pred_sales_radio']
train['residual_news'] = train['sales'] - train['pred_sales_news']

# Plotting residuals
fig, axes = plt.subplots(1, 3, figsize=(18, 6), sharex=True, sharey=True)

# TV Model
axes[0].scatter(train['sales'], train['residual_tv'], alpha=0.6)
axes[0].axhline(0, color='red', linestyle='--')
axes[0].set_title('TV Model')
axes[0].set_xlabel('Actual Sales')
axes[0].set_ylabel('Residuals')

# Radio Model
axes[1].scatter(train['sales'], train['residual_radio'], alpha=0.6)
axes[1].axhline(0, color='red', linestyle='--')
axes[1].set_title('Radio Model')
axes[1].set_xlabel('Actual Sales')
axes[1].set_ylabel('Residuals')

# Newspaper Model
axes[2].scatter(train['sales'], train['residual_news'], alpha=0.6)
axes[2].axhline(0, color='red', linestyle='--')
axes[2].set_title('Newspaper Model')
axes[2].set_xlabel('Actual Sales')
axes[2].set_ylabel('Residuals')

plt.tight_layout()
plt.show()
```

[98]: <matplotlib.collections.PathCollection at 0x7a16146b3100>

[98]: <matplotlib.lines.Line2D at 0x7a16146451e0>

[98]: Text(0.5, 1.0, 'TV Model')

[98]: Text(0.5, 0, 'Actual Sales')

[98]: Text(0, 0.5, 'Residuals')

[98]: <matplotlib.collections.PathCollection at 0x7a16145f7370>

[98]: <matplotlib.lines.Line2D at 0x7a16147c85e0>

[98]: Text(0.5, 1.0, 'Radio Model')

[98]: Text(0.5, 0, 'Actual Sales')

[98]: Text(0, 0.5, 'Residuals')

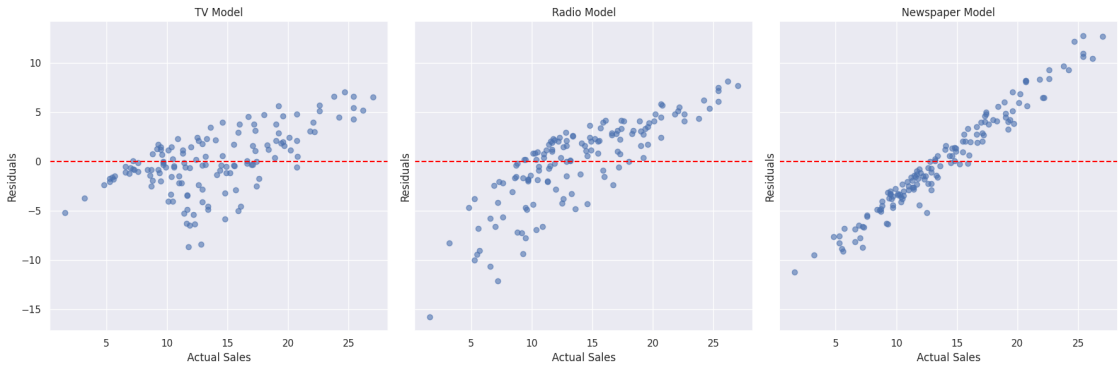
[98]: <matplotlib.collections.PathCollection at 0x7a16145f46d0>

[98]: <matplotlib.lines.Line2D at 0x7a16144f6f20>

[98]: Text(0.5, 1.0, 'Newspaper Model')

[98]: Text(0.5, 0, 'Actual Sales')

[98]: Text(0, 0.5, 'Residuals')



1.0.6 Comment

Based on the points on the graph, there seems to be some pattern in the pattern. As sales increase, the residuals also gradually increase. This implies that our model needs additional features for training.

[99]:
```
train
```

[99]:

	TV	radio	newspaper	sales	pred_sales_tv	pred_sales_radio	\
134	219.8	33.5	45.1	19.6	17.623335	16.053516	
137	25.6	39.0	9.3	9.5	7.982337	17.211943	
155	187.8	21.1	9.5	15.6	16.034705	13.441789	
178	170.2	7.8	35.2	11.7	15.160958	10.640501	
69	237.4	27.5	11.0	18.9	18.497082	14.789777	
..	…	…	…	…	…	…	
57	7.3	28.1	41.4	5.5	7.073839	14.916151	
183	56.2	5.7	29.7	8.7	9.501464	10.198193	
200	232.1	8.6	8.7	13.4	18.233965	10.809000	

```
93   217.7   33.5       59.0   19.4      17.519081         16.053516
127    7.8   38.9       50.6    6.6       7.098661         17.190881

     pred_sales_news  residual_tv  residual_radio  residual_news
134         14.476058     1.976665        3.546484       5.123942
137         12.808692     1.517663       -7.711943      -3.308692
155         12.818007    -0.434705        2.158211       2.781993
178         14.014971    -3.460958        1.059499      -2.314971
69          12.887869     0.402918        4.110223       6.012131
..                …            …               …              …
57          14.303732    -1.573839       -9.416151      -8.803732
183         13.758811    -0.801464       -1.498193      -5.058811
200         12.780747    -4.833965        2.591000       0.619253
93          15.123443     1.880919        3.346484       4.276557
127         14.732217    -0.498661      -10.590881      -8.132217

[140 rows x 10 columns]
```

Plot 3: For each of the three regression models AND each of the three features, create a scatter plot with the residuals ($y - \hat{y}$) on the vertical axis, and the feature (x) on the horizontal axis. This plot will include nine subplots in total, for every combination of regression model and feature. Use the same vertical scale for all subplots (but the horizontal scale will depend on the feature!) Make sure to clearly label each axis, and also label each subplot with a title that indicates which regression model it uses. Is there a pattern in the residuals (and if so, what might it indicate), or do they appear to have no pattern with respect to each of the three features?

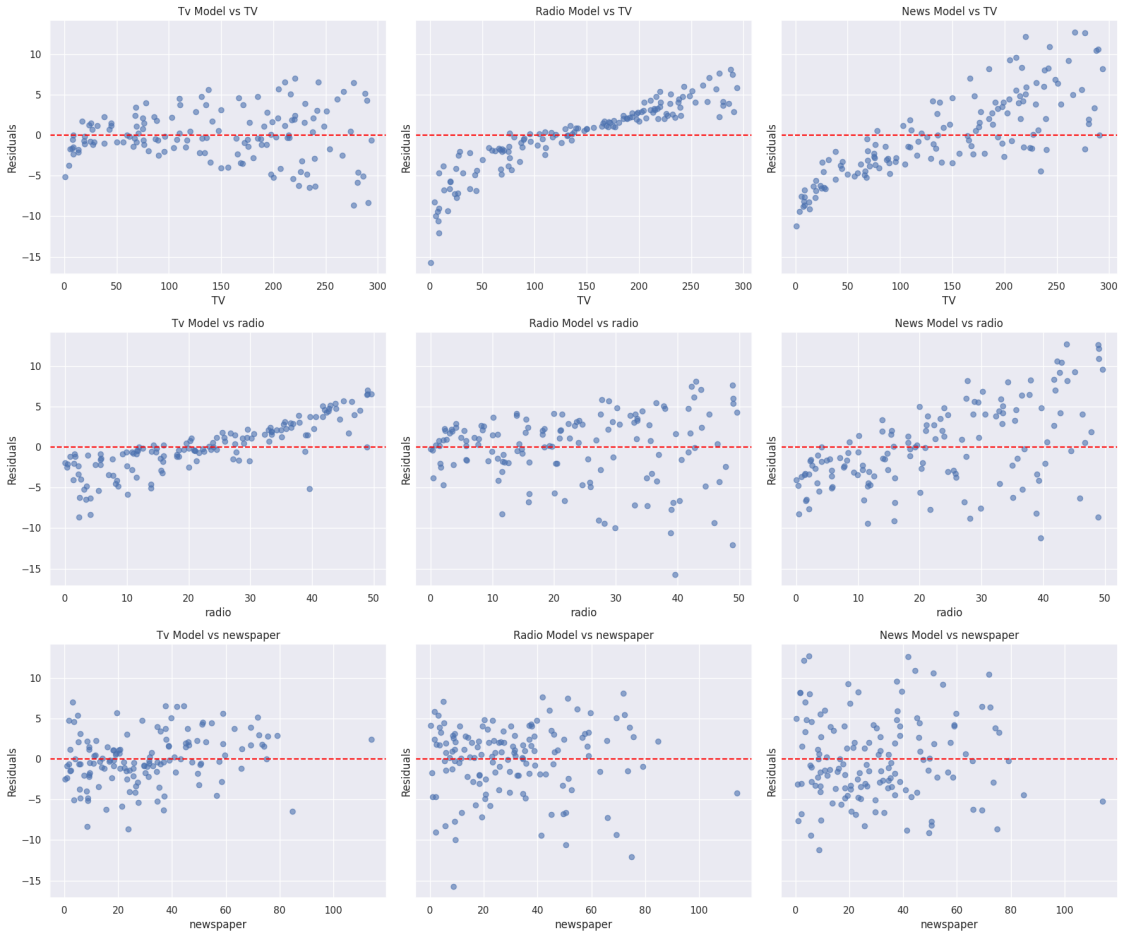
[100]:
```python
# Plotting residuals against each feature
fig, axes = plt.subplots(3, 3, figsize=(18, 15), sharey=True)

features = ['TV', 'radio', 'newspaper']
models = ['tv', 'radio', 'news']

for i, feature in enumerate(features):
    for j, model in enumerate(models):
        residuals_column = f"residual_{model}"
        axes[i, j].scatter(train[feature], train[residuals_column], alpha=0.6)
        axes[i, j].axhline(0, color='red', linestyle='--')
        axes[i, j].set_title(f'{model.capitalize()} Model vs {feature}')
        axes[i, j].set_xlabel(feature)
        axes[i, j].set_ylabel('Residuals')

plt.tight_layout()
plt.show()
```

[100]:
```
<matplotlib.collections.PathCollection at 0x7a160df6fa00>
```

[100]:
```
<matplotlib.lines.Line2D at 0x7a160df6fe20>
```

```
[100]: Text(0.5, 1.0, 'Tv Model vs TV')

[100]: Text(0.5, 0, 'TV')

[100]: Text(0, 0.5, 'Residuals')

[100]: <matplotlib.collections.PathCollection at 0x7a160df9c430>

[100]: <matplotlib.lines.Line2D at 0x7a160df9c220>

[100]: Text(0.5, 1.0, 'Radio Model vs TV')

[100]: Text(0.5, 0, 'TV')

[100]: Text(0, 0.5, 'Residuals')

[100]: <matplotlib.collections.PathCollection at 0x7a160df9ce80>

[100]: <matplotlib.lines.Line2D at 0x7a160df9cc70>

[100]: Text(0.5, 1.0, 'News Model vs TV')

[100]: Text(0.5, 0, 'TV')

[100]: Text(0, 0.5, 'Residuals')

[100]: <matplotlib.collections.PathCollection at 0x7a160df9d7b0>

[100]: <matplotlib.lines.Line2D at 0x7a160df9d600>

[100]: Text(0.5, 1.0, 'Tv Model vs radio')

[100]: Text(0.5, 0, 'radio')

[100]: Text(0, 0.5, 'Residuals')

[100]: <matplotlib.collections.PathCollection at 0x7a160df9e0e0>

[100]: <matplotlib.lines.Line2D at 0x7a1614d86350>

[100]: Text(0.5, 1.0, 'Radio Model vs radio')

[100]: Text(0.5, 0, 'radio')

[100]: Text(0, 0.5, 'Residuals')

[100]: <matplotlib.collections.PathCollection at 0x7a160df9ea10>
```

```
[100]: <matplotlib.lines.Line2D at 0x7a1614daae00>

[100]: Text(0.5, 1.0, 'News Model vs radio')

[100]: Text(0.5, 0, 'radio')

[100]: Text(0, 0.5, 'Residuals')

[100]: <matplotlib.collections.PathCollection at 0x7a160df9f2b0>

[100]: <matplotlib.lines.Line2D at 0x7a160df9f0a0>

[100]: Text(0.5, 1.0, 'Tv Model vs newspaper')

[100]: Text(0.5, 0, 'newspaper')

[100]: Text(0, 0.5, 'Residuals')

[100]: <matplotlib.collections.PathCollection at 0x7a160df9fbe0>

[100]: <matplotlib.lines.Line2D at 0x7a160df6fd30>

[100]: Text(0.5, 1.0, 'Radio Model vs newspaper')

[100]: Text(0.5, 0, 'newspaper')

[100]: Text(0, 0.5, 'Residuals')

[100]: <matplotlib.collections.PathCollection at 0x7a160df9fb50>

[100]: <matplotlib.lines.Line2D at 0x7a1614e54e50>

[100]: Text(0.5, 1.0, 'News Model vs newspaper')

[100]: Text(0.5, 0, 'newspaper')

[100]: Text(0, 0.5, 'Residuals')
```



1.0.7 Comment

In the graphs "TV model vs Radio," "Radio model vs TV," and "News model vs TV," there are noticeable pattern. As the corresponding advertising investment increases, the residuals show a more apparent upward trend. This suggests that for these models, more factors are needed for training to eliminate the existing patterns.

1.0.8 3. Try a multiple linear regression

Next, fit a multiple linear regression to predict product sales, using all three features to train a single model: TV ad budget, radio ad budget, and newspaper ad budget.

Print the intercept and coefficients, and compute the MSE and R2 on the training data, and MSE and R2 on the test data. Comment on the results. Make sure to explain any differences between the coefficients of the multiple regression model, and the coefficients of the three simple linear regression models. If they are different, why?

The code in the first part of this section is provided for you. However, you will need to add comments, observations, and answers to the questions.

Also repeat the analysis of part (3) for this regression model. Use training data for all of these items:

Plot 1: Create a scatter plot of predicted sales ($\hat{y}$) on the vertical axis, and actual sales (y) on the horizontal axis. Make sure both axes use the same scale (the range of the vertical axis should be the same as the range of the horizontal axis). Label each axes. Does this model explain the data more effectively than the simple linear regressions from the previous section?

Plot 2: Create a scatter plot with the residuals ($y - \hat{y}$) on the vertical axis, and actual sales (y) on the horizontal axis. Comment on your observations. Is there a pattern in the residuals (and if so, what might it indicate), or do they appear to have no pattern with respect to actual sales?

Plot 3: For each of the three features, plot the residuals ($y - \hat{y}$) on the vertical axis, and the feature (x) on the horizontal axis. Make sure to clearly label each axis. Is there a pattern in the residuals (and if so, what might it indicate), or do they appear to have no pattern with respect to each of the three features?

Note that in general, to earn full credit, plots must:

- Be readable (especially text size).
- Have a label on each axis.
- Have an appropriate range for each axis. When there are multiple subplots, if the goal is to compare similar things in different subplots, in most cases it is appropriate for them all to use the same range.
- If there are multiple subplots, or multiple data series in the same plot, it must be made clear which is which.

Fit a multiple linear regression

[101]:
```
reg_multi = LinearRegression().fit(train[['TV', 'radio', 'newspaper']],␣
↪train['sales'])
```

Look at coefficients

[102]:
```
print("Coefficients (TV, radio, newspaper):", reg_multi.coef_)
print("Intercept: ", reg_multi.intercept_)
```

```
Coefficients (TV, radio, newspaper): [ 0.04636712  0.18249225 -0.00196151]
Intercept:  3.0762941463550604
```

Compute R2, MSE for multiple regression

[103]:
```
y_pred_tr_multi = reg_multi.predict(train[['TV', 'radio', 'newspaper']])

r2_tr_multi  = metrics.r2_score(train['sales'], y_pred_tr_multi)
mse_tr_multi = metrics.mean_squared_error(train['sales'], y_pred_tr_multi)

print("Multiple regression R2 of Train:  ", r2_tr_multi)
print("Multiple regression MSE of Train: ", mse_tr_multi)
```

```
Multiple regression R2 of Train:   0.8934006397815405
Multiple regression MSE of Train:  2.952755722412376
```

[104]:
```
y_pred_ts_multi = reg_multi.predict(test[['TV', 'radio', 'newspaper']])

r2_ts_multi  = metrics.r2_score(test['sales'], y_pred_ts_multi)
mse_ts_multi = metrics.mean_squared_error(test['sales'], y_pred_ts_multi)

print("Multiple regression R2 of Test:  ", r2_ts_multi)
print("Multiple regression MSE of Test: ", mse_ts_multi)
```

```
Multiple regression R2 of Test:   0.9034495005656622
Multiple regression MSE of Test:  2.4406300760885373
```

1.0.9 Comment

From the R2 (0.9034) and MSE (2.4406) of the multiple model, it can be concluded: Since the R2 of the multiple model is larger than that of the model trained with a single element, and the MSE is smaller than that of the model trained with a single element, the multiple model can explain the data more effectively compared to the simple model.

[105]:
```
### Plot 1 ###

# Calculating predicted sales for multiple regression model
train['pred_sales_multiple'] = reg_multi.
 ↪predict(train[['TV','radio','newspaper']])

# Plotting again
fig, ax = plt.subplots(figsize=(6, 6))

# Multiple Model
ax.scatter(train['sales'], train['pred_sales_multiple'], alpha=0.6)
ax.plot([0, 30], [0, 30], '--', color='red')
ax.set_title('Multiple Model')
ax.set_xlabel('Actual Sales')
ax.set_ylabel('Predicted Sales')

plt.tight_layout()
plt.show()
```

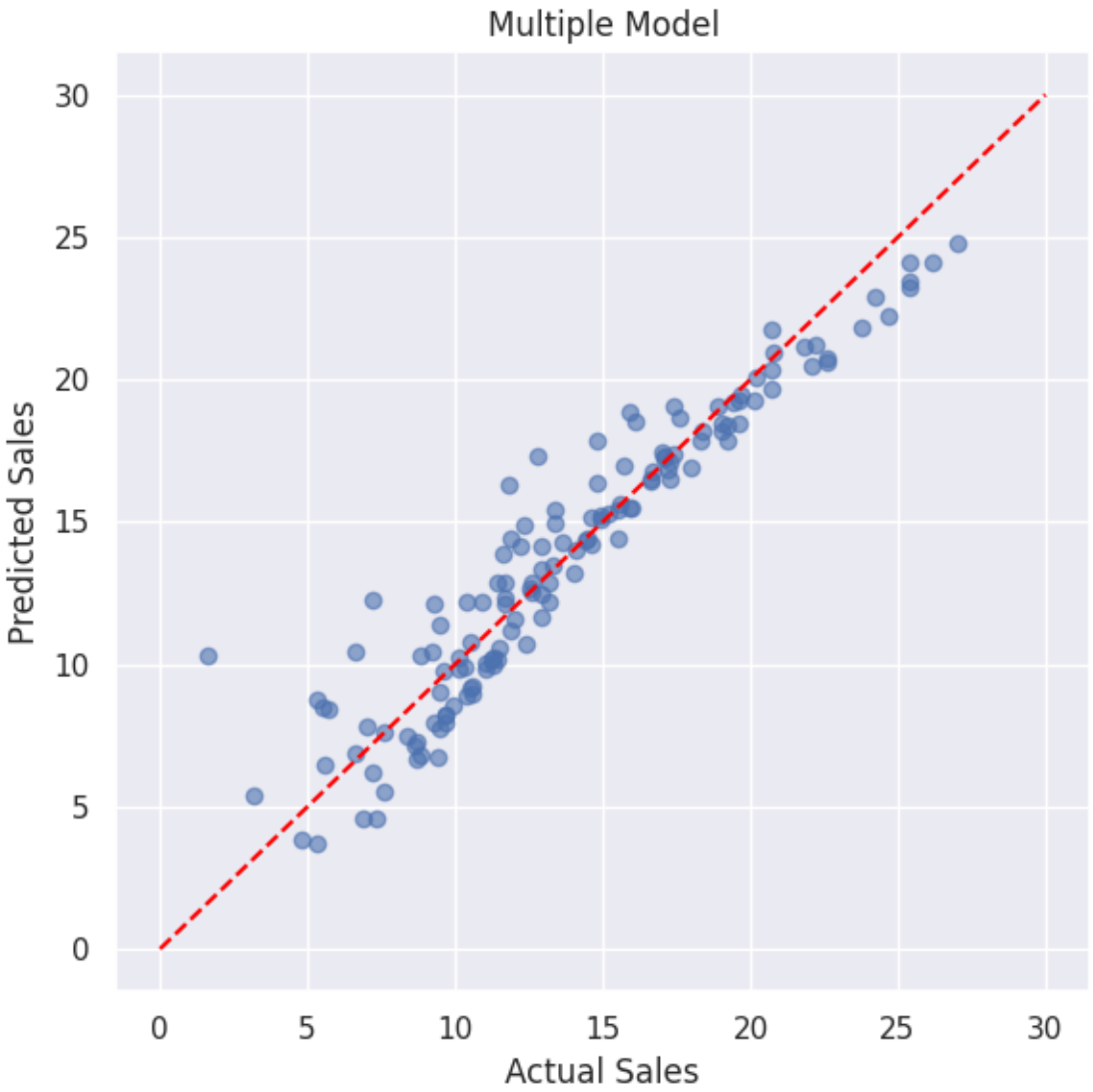
[105]: <matplotlib.collections.PathCollection at 0x7a160dd8b7f0>

[105]: [<matplotlib.lines.Line2D at 0x7a160dd8bc70>]

[105]: Text(0.5, 1.0, 'Multiple Model')

[105]: Text(0.5, 0, 'Actual Sales')

[105]: Text(0, 0.5, 'Predicted Sales')



1.0.10 Comment

From the graph, it can be seen that the points are closer to the red dashed line y = x. Therefore, the multiple model can explain the data more effectively.

[106]:
```
### Plot 2 ###

# Calculating residuals for multiple regression model
train['residual_multiple'] = train['sales'] - train['pred_sales_multiple']

# Plotting again
```

```
fig, ax = plt.subplots(figsize=(6, 6))

# Multiple Model
ax.scatter(train['sales'], train['residual_multiple'], alpha=0.6)
ax.axhline(0, color='red', linestyle='--')

# Set y-axis limits to ensure the red dashed line is in the middle
max_residual = abs(train['residual_multiple']).max()
ax.set_ylim(-max_residual, max_residual)

ax.set_title('Multiple Model')
ax.set_xlabel('Actual Sales')
ax.set_ylabel('Residuals')

plt.tight_layout()
plt.show()
```

```
[106]: <matplotlib.collections.PathCollection at 0x7a160dc28760>

[106]: <matplotlib.lines.Line2D at 0x7a160dc05f90>

[106]: (-8.718378893529223, 8.718378893529223)

[106]: Text(0.5, 1.0, 'Multiple Model')

[106]: Text(0.5, 0, 'Actual Sales')

[106]: Text(0, 0.5, 'Residuals')
```

1.0.11 Comment

The points in the graph seem to follow a certain distribution pattern, that is, as Sales increase, Residuals first decrease and then increase, indicating that there is a pattern in the residuals.

[107]:
```
### Plot 3 ###

# Plotting residuals
fig, axes = plt.subplots(1, 3, figsize=(18, 6), sharex=True,sharey=True)

# Set y-axis limits to ensure the red dashed line is in the middle
max_residual = abs(train['residual_multiple']).max()
```

```
# TV Model
axes[0].scatter(train['TV'], train['residual_multiple'], alpha=0.6)
axes[0].set_ylim(-max_residual, max_residual)
axes[0].axhline(0, color='red', linestyle='--')
axes[0].set_title('TV Model')
axes[0].set_xlabel('TV')
axes[0].set_ylabel('Residuals')

# Radio Model
axes[1].scatter(train['radio'], train['residual_multiple'], alpha=0.6)
axes[1].axhline(0, color='red', linestyle='--')
axes[1].set_title('Radio Model')
axes[1].set_xlabel('Radio')
axes[1].set_ylabel('Residuals')

# Newspaper Model
axes[2].scatter(train['newspaper'], train['residual_multiple'], alpha=0.6)
axes[2].axhline(0, color='red', linestyle='--')
axes[2].set_title('Newspaper Model')
axes[2].set_xlabel('Newspaper')
axes[2].set_ylabel('Residuals')

plt.tight_layout()
plt.show()
```

[107]: <matplotlib.collections.PathCollection at 0x7a160dcf4310>

[107]: (-8.718378893529223, 8.718378893529223)

[107]: <matplotlib.lines.Line2D at 0x7a160dcc4490>

[107]: Text(0.5, 1.0, 'TV Model')

[107]: Text(0.5, 0, 'TV')

[107]: Text(0, 0.5, 'Residuals')

[107]: <matplotlib.collections.PathCollection at 0x7a160db24e80>

[107]: <matplotlib.lines.Line2D at 0x7a161122c130>

[107]: Text(0.5, 1.0, 'Radio Model')

[107]: Text(0.5, 0, 'Radio')

[107]: Text(0, 0.5, 'Residuals')

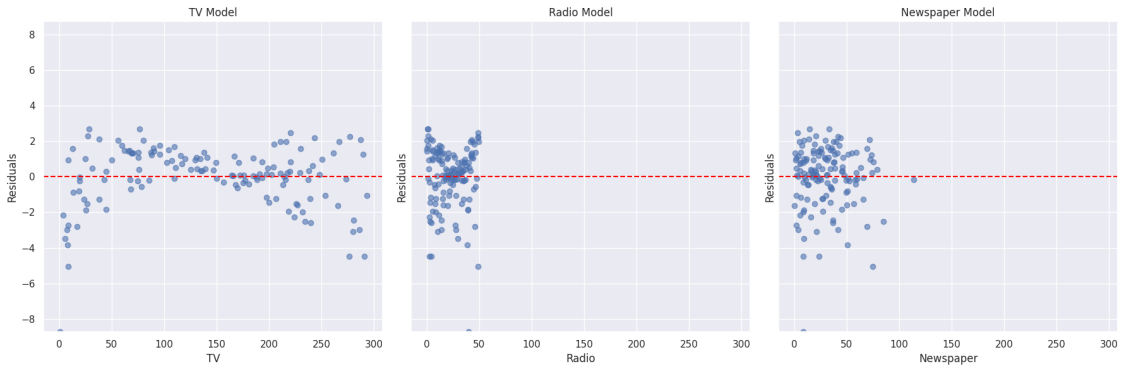
[107]: <matplotlib.collections.PathCollection at 0x7a160db258d0>

[107]: <matplotlib.lines.Line2D at 0x7a160dcc5ed0>

[107]: Text(0.5, 1.0, 'Newspaper Model')

[107]: Text(0.5, 0, 'Newspaper')

[107]: Text(0, 0.5, 'Residuals')



1.0.12 Comment

For these three graphs, there doesn't seem to be any obvious pattern, that is, the points in the graphs are basically randomly distributed along the red dashed line where the residuals are zero.

[108]:
```
train[['TV', 'radio', 'newspaper']].corr()
```

[108]:

	TV	radio	newspaper
TV	1.000000	0.106568	0.057997
radio	0.106568	1.000000	0.314422
newspaper	0.057997	0.314422	1.000000

1.0.13 4. Linear regression with interaction terms

Our multiple linear regression includes additive effects of all three types of advertising media. However, it does not include *interaction* effects, in which combining different types of advertising media together results in a bigger boost in sales than just the additive effect of the individual media.

The pattern in the residuals plots from parts (1) through (3) suggest that a model including an interaction effect may explain sales data better than a model including additive effects. Add four columns to each data frame (`train` and `test`):

- `newspaper` × `radio` (name this column `newspaper_radio`)
- `TV` × `radio` (name this column `TV_radio`)
- `newspaper` × `TV` (name this column `newspaper_TV`)

- `newspaper` × `radio` × `TV` (name this column `newspaper_radio_TV`)

Note: you can use the `assign` function in `pandas` (documentation here) to create a new column and assign a value to it using operations on other columns.

Then, train a linear regression model on all seven features: the three types of ad budgets, and the four interaction effects. Repeat the analysis of part (3) for the model including interaction effects. Are the interaction effects helpful for explaining the effect of ads on product sales? Are there any patterns evident in the residual plots that suggest further opportunities for improving the model?

The code in this section is not provided for you. You will need to write code, in addition to comments, observations, and answers to the questions.

Note that in general, to earn full credit, plots must:

- Be readable (especially text size).
- Have a label on each axis.
- Have an appropriate range for each axis. When there are multiple subplots, if the goal is to compare similar things in different subplots, in most cases it is appropriate for them all to use the same range.
- If there are multiple subplots, or multiple data series in the same plot, it must be made clear which is which.

[109]:
```
### reload the data ###
data = pd.read_csv('Advertising.csv', index_col=0)

### Adding interaction terms to the dataset ###
data=data.assign(
    newspaper_radio = data["newspaper"] * data["radio"],
    TV_radio = data["TV"] * data["radio"],
    newspaper_TV = data["newspaper"] * data["TV"],
    newspaper_radio_TV = data["newspaper"] * data["radio"] * data["TV"]
)
data
```

[109]:

	TV	radio	newspaper	sales	newspaper_radio	TV_radio	newspaper_TV	\
1	230.1	37.8	69.2	22.1	2615.76	8697.78	15922.92	
2	44.5	39.3	45.1	10.4	1772.43	1748.85	2006.95	
3	17.2	45.9	69.3	9.3	3180.87	789.48	1191.96	
4	151.5	41.3	58.5	18.5	2416.05	6256.95	8862.75	
5	180.8	10.8	58.4	12.9	630.72	1952.64	10558.72	
..	…	…	…	…	…	…	…	
196	38.2	3.7	13.8	7.6	51.06	141.34	527.16	
197	94.2	4.9	8.1	9.7	39.69	461.58	763.02	
198	177.0	9.3	6.4	12.8	59.52	1646.10	1132.80	
199	283.6	42.0	66.2	25.5	2780.40	11911.20	18774.32	
200	232.1	8.6	8.7	13.4	74.82	1996.06	2019.27	

	newspaper_radio_TV

```
1            601886.376
2             78873.135
3             54710.964
4            366031.575
5            114034.176
..                  …
196            1950.492
197            3738.798
198           10535.040
199          788521.440
200           17365.722

[200 rows x 8 columns]
```

[110]:
```
# Features for the regression model
features = ["TV", "radio", "newspaper", "newspaper_radio", "TV_radio",␣
 ↪"newspaper_TV", "newspaper_radio_TV"]
```

Fit a interaction linear regression

[111]:
```
train, test = train_test_split(data, test_size=0.3, random_state=9)

reg_inter = LinearRegression().fit(train[features], train['sales'])
```

Look at coefficients

[112]:
```
print("Coefficients (interaction terms):", reg_inter.coef_)
print("Intercept: ", reg_inter.intercept_)
```

```
Coefficients (interaction terms): [ 2.07581277e-02  1.19519832e-02
2.03755245e-02 -2.46384818e-05
  1.18859985e-03 -8.06743937e-05 -7.25626214e-07]
Intercept:  6.378007972477893
```

Compute R2, MSE for interaction regression

[113]:
```
y_pred_tr_inter = reg_inter.predict(train[features])

r2_tr_inter  = metrics.r2_score(train['sales'], y_pred_tr_inter)
mse_tr_inter = metrics.mean_squared_error(train['sales'], y_pred_tr_inter)

print("Interaction regression R2:  ", r2_tr_inter)
print("Interaction regression MSE: ", mse_tr_inter)
```

```
Interaction regression R2:   0.9639737928022052
Interaction regression MSE:  0.997910205484344
```

[114]:
```
y_pred_ts_inter = reg_inter.predict(test[features])

r2_ts_inter  = metrics.r2_score(test['sales'], y_pred_ts_inter)
mse_ts_inter = metrics.mean_squared_error(test['sales'], y_pred_ts_inter)

print("Interaction regression R2:  ", r2_ts_inter)
print("Interaction regression MSE: ", mse_ts_inter)
```

```
Interaction regression R2:   0.978290346306146
Interaction regression MSE:  0.5487825962280087
```

1.0.14 Comment

The Interaction regression model has a larger R2 and a smaller MSE compared to the multiple model. This means that the Interaction model can explain the data more effectively.

[115]:
```
### Plot 1 ###

# Calculating predicted sales for multiple regression model
train['pred_sales_inter'] = reg_inter.predict(train[features])

# Plotting again
fig, ax = plt.subplots(figsize=(6, 6))

# Multiple Model
ax.scatter(train['sales'], train['pred_sales_inter'], alpha=0.6)
ax.plot([0, 30], [0, 30], '--', color='red')
ax.set_title('Interaction Model')
ax.set_xlabel('Actual Sales')
ax.set_ylabel('Predicted Sales')

plt.tight_layout()
plt.show()
```

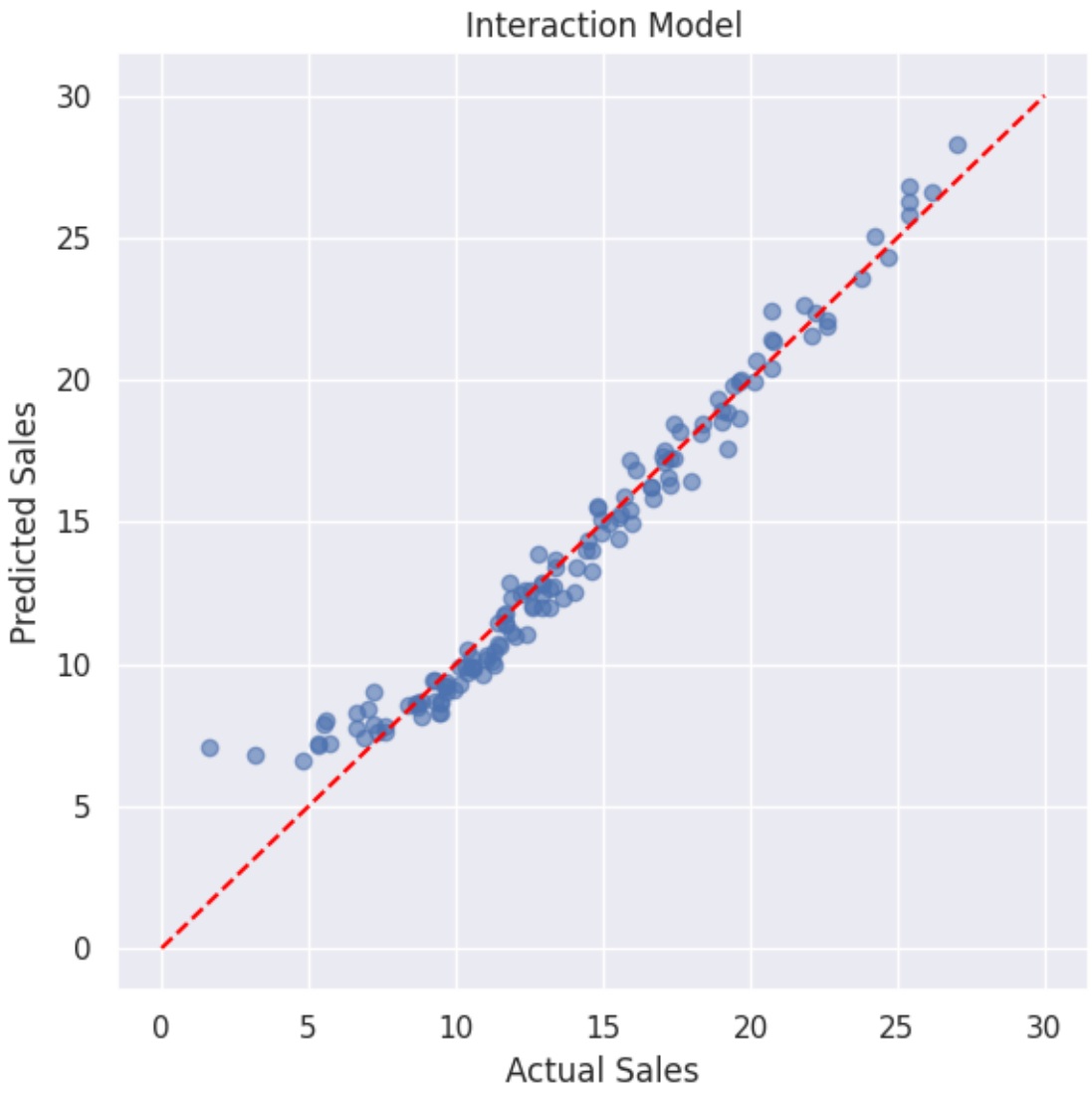
[115]: <matplotlib.collections.PathCollection at 0x7a1617023a30>

[115]: [<matplotlib.lines.Line2D at 0x7a164fb81330>]

[115]: Text(0.5, 1.0, 'Interaction Model')

[115]: Text(0.5, 0, 'Actual Sales')

[115]: Text(0, 0.5, 'Predicted Sales')



1.0.15 Comment

Compared to the multiple model, the points in the interaction regression model's graph are closer to y = x. This also proves that the interaction model can explain the data more effectively.

[116]:
```
### Plot 2 ###

# Calculating residuals for multiple regression model
train['residual_interaction'] = train['sales'] - train['pred_sales_inter']

# Plotting again
fig, ax = plt.subplots(figsize=(6, 6))
```

```
# Multiple Model
ax.scatter(train['sales'], train['residual_interaction'], alpha=0.6)
ax.axhline(0, color='red', linestyle='--')

# Set y-axis limits to ensure the red dashed line is in the middle
max_residual = abs(train['residual_interaction']).max()
ax.set_ylim(-max_residual, max_residual)

ax.set_title('Interaction Model')
ax.set_xlabel('Actual Sales')
ax.set_ylabel('Residuals')

plt.tight_layout()
plt.show()
```

[116]: <matplotlib.collections.PathCollection at 0x7a16146454b0>

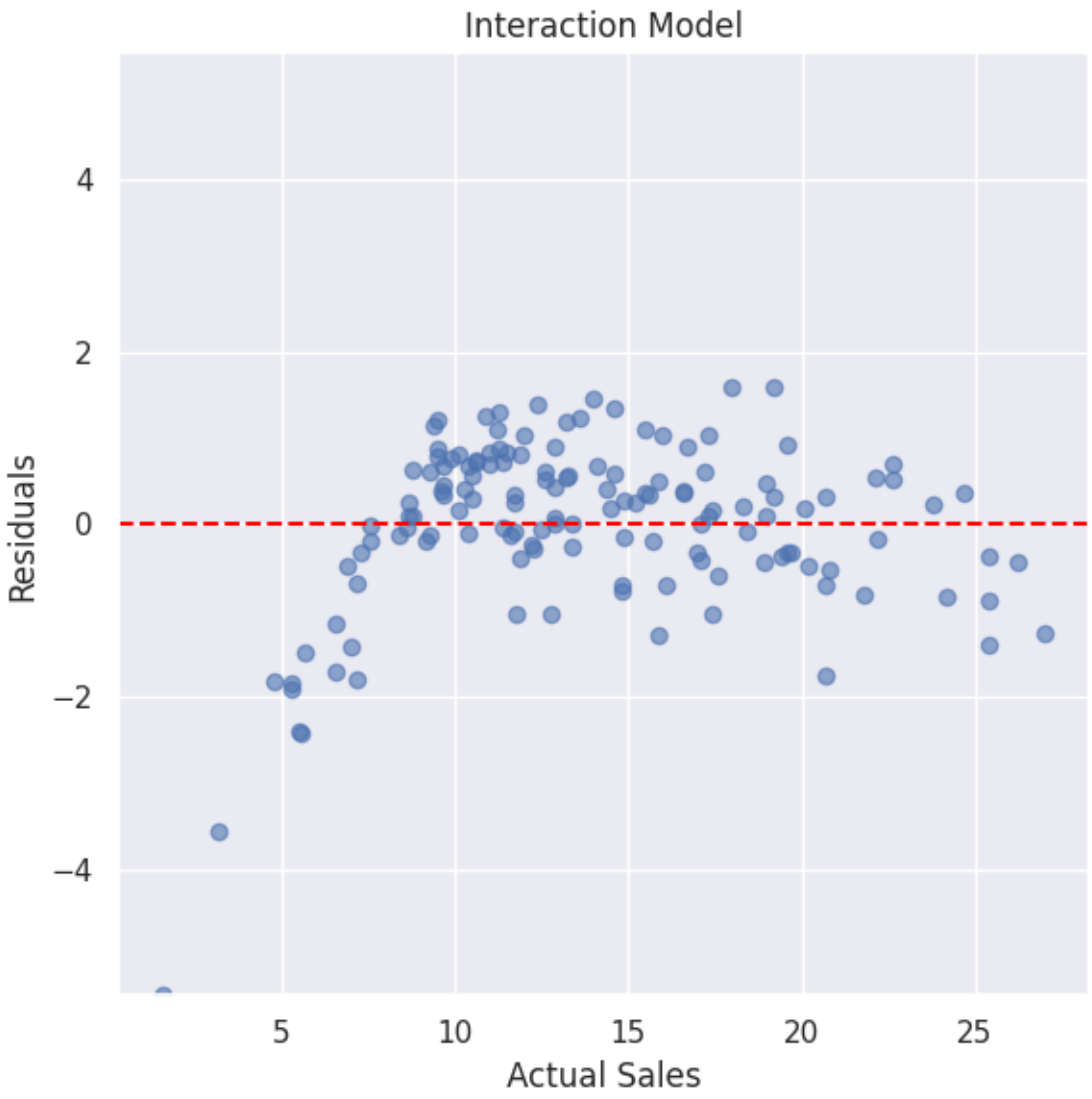
[116]: <matplotlib.lines.Line2D at 0x7a161458e200>

[116]: (-5.466897496905435, 5.466897496905435)

[116]: Text(0.5, 1.0, 'Interaction Model')

[116]: Text(0.5, 0, 'Actual Sales')

[116]: Text(0, 0.5, 'Residuals')



1.0.16 Comment

The distribution of points in the graph follows a certain pattern, that is, as Actual Sales increase, residuals first increase and then decrease. This implies that there is still a certain pattern present in the model.

[117]:
```
### Plot 3 ###

# Plotting residuals
fig, axes = plt.subplots(1, 3, figsize=(18, 6), sharex=True,sharey=True)

# Set y-axis limits to ensure the red dashed line is in the middle
max_residual = abs(train['residual_interaction']).max()
```

```
# TV Model
axes[0].scatter(train['TV'], train['residual_interaction'], alpha=0.6)
axes[0].set_ylim(-max_residual, max_residual)
axes[0].axhline(0, color='red', linestyle='--')
axes[0].set_title('TV Model')
axes[0].set_xlabel('TV')
axes[0].set_ylabel('Residuals')

# Radio Model
axes[1].scatter(train['radio'], train['residual_interaction'], alpha=0.6)
axes[1].axhline(0, color='red', linestyle='--')
axes[1].set_title('Radio Model')
axes[1].set_xlabel('Radio')
axes[1].set_ylabel('Residuals')

# Newspaper Model
axes[2].scatter(train['newspaper'], train['residual_interaction'], alpha=0.6)
axes[2].axhline(0, color='red', linestyle='--')
axes[2].set_title('Newspaper Model')
axes[2].set_xlabel('Newspaper')
axes[2].set_ylabel('Residuals')

plt.tight_layout()
plt.show()
```

[117]: <matplotlib.collections.PathCollection at 0x7a160d90e710>

[117]: (-5.466897496905435, 5.466897496905435)

[117]: <matplotlib.lines.Line2D at 0x7a160d90edd0>

[117]: Text(0.5, 1.0, 'TV Model')

[117]: Text(0.5, 0, 'TV')

[117]: Text(0, 0.5, 'Residuals')

[117]: <matplotlib.collections.PathCollection at 0x7a160d9433d0>

[117]: <matplotlib.lines.Line2D at 0x7a160df9f1c0>

[117]: Text(0.5, 1.0, 'Radio Model')

[117]: Text(0.5, 0, 'Radio')

[117]: Text(0, 0.5, 'Residuals')

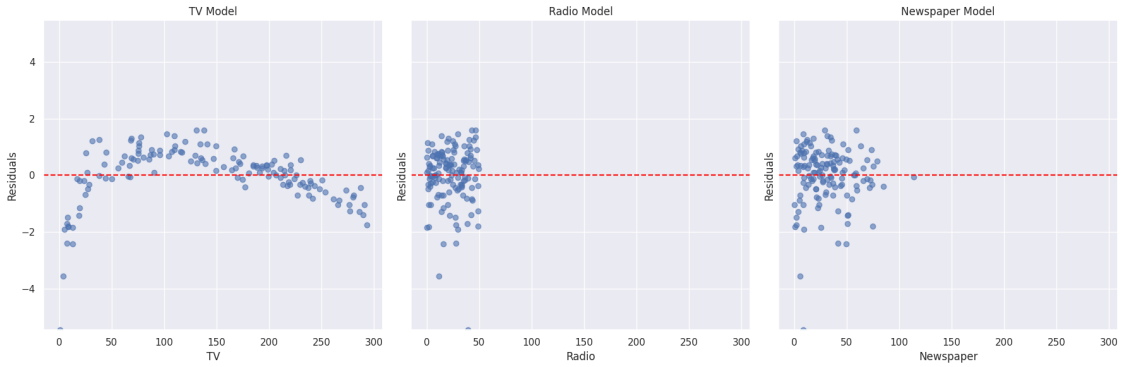
[117]: <matplotlib.collections.PathCollection at 0x7a160d943b20>

[117]: <matplotlib.lines.Line2D at 0x7a161175f430>

[117]: Text(0.5, 1.0, 'Newspaper Model')

[117]: Text(0.5, 0, 'Newspaper')

[117]: Text(0, 0.5, 'Residuals')



1.0.17 Comment

In the "TV model" graph, the points display a very obvious pattern. Specifically, as the investment in TV advertising increases, the residuals first increase and then decrease. This implies that there is a pattern in the model.