Agustin Leon Nunez - al8937

Assignment: Linear regression on the Advertising data

Fraida Fund

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.

In [91]:

```
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"
```

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

In [92]:

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

```
--2024-09-30 02:59:15--  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... connect
ed.
HTTP request sent, awaiting response... 302 Found
Location: https://static1.squarespace.com/static/5ff2adbe3fe4fe33db902812/t/5fffe03b40910
76ff5b30c72/1610604603901/Advertising.csv [following]
--2024-09-30 02:59:15--  https://static1.squarespace.com/static/5ff2adbe3fe4fe33db902812/
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... con
nected.
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

2024-09-30 02:59:15 (41.7 MB/s) - ‘Advertising.csv’ saved [4555/4555]
```

In [93]:

```
df  = pd.read_csv('Advertising.csv', index_col=0)
df.head()
```

Out[93]:

	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

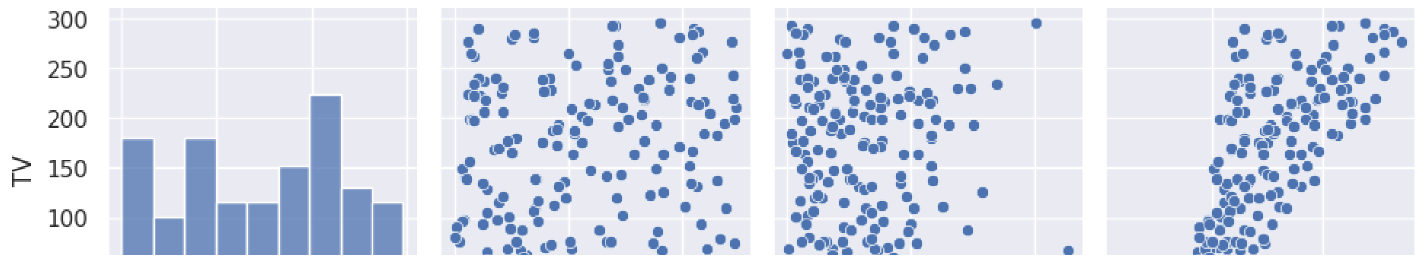
Note that in this dataset, the first column in the data file is the row label; that’s why we use `index_col=0` in the `read_csv` command. If we would omit that argument, then we would have an additional (unnamed) column in the dataset, containing the row number.

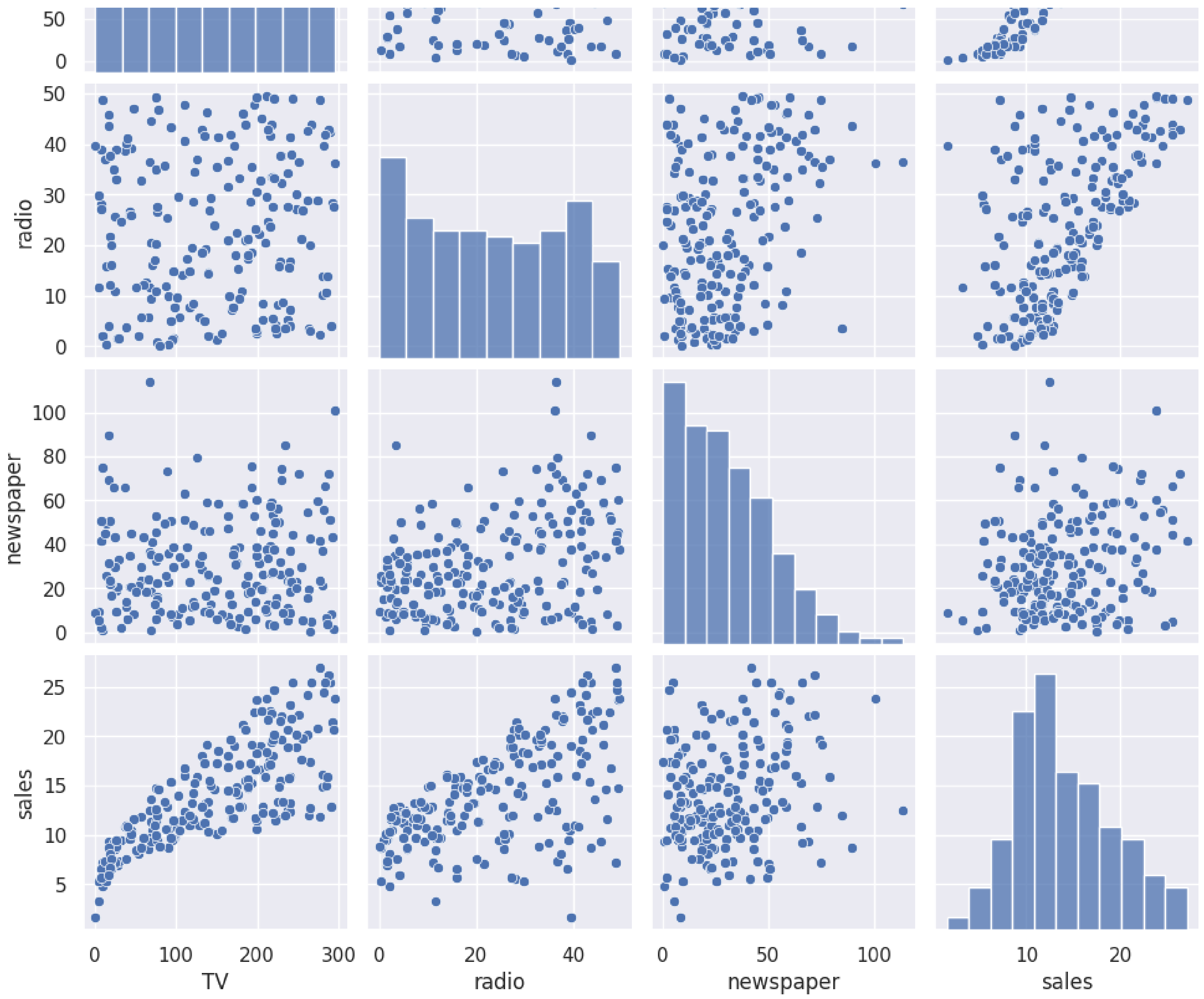
(You can try removing the `index_col` argument and re-running the cell above, to see the effect and to understand why we used this argument.)

Visually inspect the data

In [94]:

```
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.

In [95]:

```
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.

In [96]:

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 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
```

In [97]:

```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 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. 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

In [98]:

```
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

In [99]:

```
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_)
```

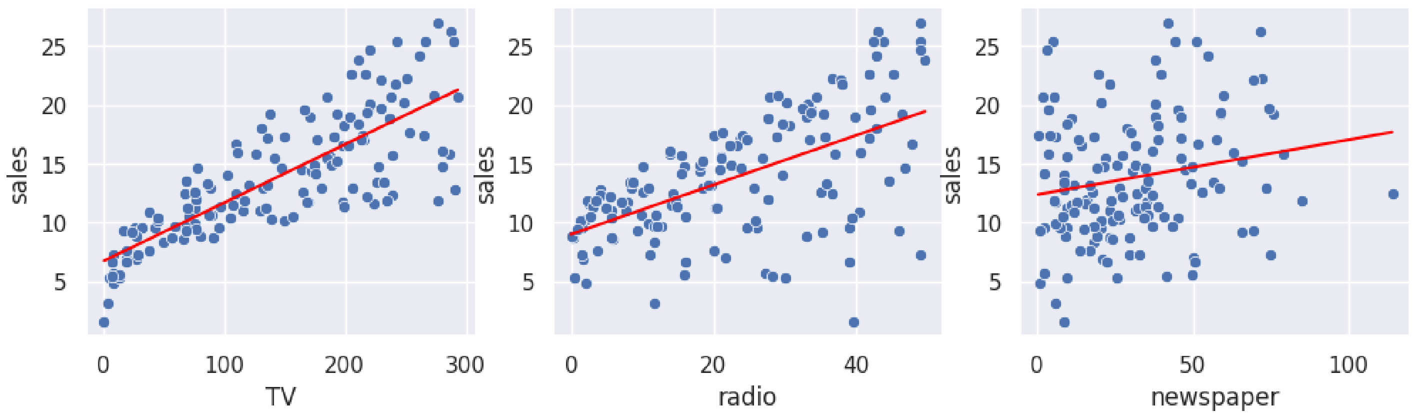
```
TV       :  0.04964468781898984 6.711432632336138
Radio    :  0.21062312839115208 8.997640913704718
Newspaper:  0.046574464282301664 12.375549417451523
```

Plot data and regression line

In [100]:

```
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

In [101]:

```
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']])
```

In [102]:

```
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
```

In [103]:

```
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
```

In [104]:

```
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']])
```

In [105]:

```
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
```

In [106]:

```
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
```

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?

Radio spending seems to be associated with the largest reward in sales, because the coefficient is the largest.

However, when comparing the R2 of each model, the one with the largest R2 is the model with the TV advertising variable. This is the most effective model at predicting product sales.

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.

Comments:

Plot 1: We would ideally plot a perfect diagonal line between the predicted values and the actual sales. This would mean that our model is perfectly predicting sales. We can see, however, that our plots don't quite resemble a perfect diagonal line. The one that comes somewhat close is the TV ads model.

Plot 2: There is definitely a pattern between the residuals and the actual sales, which means that the model is not capturing systematic features that have an impact on the sales. The model with the least degree of "systematic residuals look" is the TV Model. However, none of the models display a random relationship between the residuals and the actual sales.

Plot 3: When plotting residuals against feature variables, we expect to see a random relationship, with no clear patterns. These plots illustrate a relationship between the TV and radio advertising features. For the model that only includes TV, we see that the residuals positively correlate with the radio feature, given that it's not included in the model. This suggests that we might need to add the radio advertising variable into the model. For the plots located in the diagonal (TV Model agaisnt TV feature, Radio Model against radio feature, and so on), we see a fair degree of randomness, which is a good indicator.

In [107]:

```python
# Plot 1: scatter of predicted sales vs actual sales

fig = plt.figure(figsize=(12,3))

plt.subplot(1,3,1)
sns.scatterplot(data=train, x=train["sales"], y=y_pred_tr_tv);
plt.ylabel('predicted y (TV)');
plt.ylim(0, 25)

plt.subplot(1,3,2)
sns.scatterplot(data=train, x=train["sales"], y=y_pred_tr_radio);
plt.ylabel('predicted y (radio)');
plt.ylim(0, 25)


plt.subplot(1,3,3)
sns.scatterplot(data=train, x=train["sales"], y=y_pred_tr_news);
plt.ylabel('predicted y (news)');
plt.ylim(0, 25)
```

Out[107]:

```
<Axes: >
```

Out[107]:

```
<Axes: xlabel='sales'>
```

Out[107]:

```
Text(0, 0.5, 'predicted y (TV)')
```

Out[107]:

```
(0.0, 25.0)
```

Out[107]:

```
<Axes: >
```

Out[107]:

```
<Axes: xlabel='sales'>
```

Out[107]:

```
Text(0, 0.5, 'predicted y (radio)')
```

Out[107]:

```
(0.0, 25.0)
```

Out[107]:

```
<Axes: >
```

Out[107]:

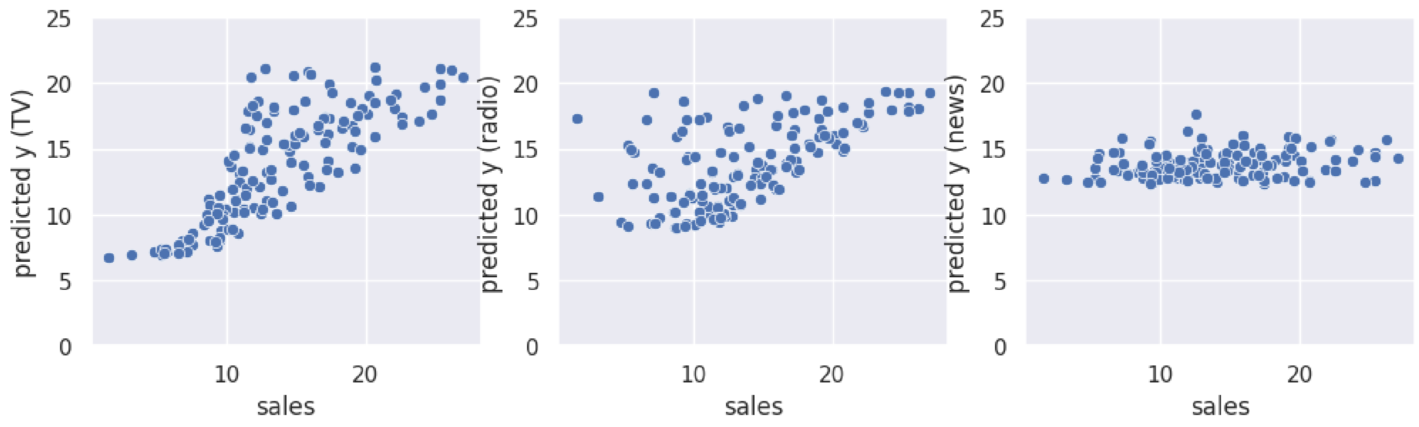
```
<Axes: xlabel='sales'>
```

Out[107]:

```
Text(0, 0.5, 'predicted y (news)')
```

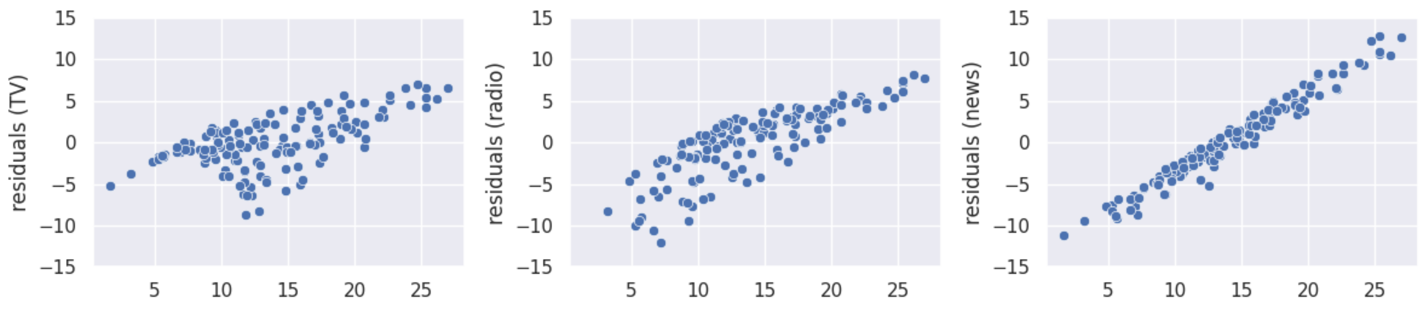
Out[107]:

```
(0.0, 25.0)
```



In [108]:

```
# Plot 2: scatter plot of the residuals vs actual sales

residual_tr_tv = train['sales'] - y_pred_tr_tv
residual_tr_radio = train['sales'] - y_pred_tr_radio
residual_tr_news = train['sales'] - y_pred_tr_news

fig = plt.figure(figsize=(12,3))

plt.subplot(1,3,1)
sns.scatterplot(data=train, x=train['sales'], y=residual_tr_tv);
plt.ylabel('residuals (TV)');
plt.ylim(-15, 15);
plt.tight_layout();

plt.subplot(1,3,2)
sns.scatterplot(data=train, x=train['sales'], y=residual_tr_radio);
plt.ylabel('residuals (radio)');
plt.ylim(-15, 15);
plt.tight_layout();


plt.subplot(1,3,3)
sns.scatterplot(data=train, x=train['sales'], y=residual_tr_news);
plt.ylabel('residuals (news)');
plt.ylim(-15, 15);
plt.tight_layout();
```



sales sales sales

In [109]:

```
# Plot 3: residuals vs feature each of the feature (nine subplots).


fig = plt.figure(figsize=(24,9))

# 1) TV model plots
plt.subplot(3,3,1)
sns.scatterplot(data=train, x=train['TV'], y=residual_tr_tv);
plt.ylabel('residuals');
plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: TV")

plt.subplot(3,3,2)
sns.scatterplot(data=train, x=train['radio'], y=residual_tr_tv);
plt.ylabel('residuals');
plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: TV")

plt.subplot(3,3,3)
sns.scatterplot(data=train, x=train['newspaper'], y=residual_tr_tv);
plt.ylabel('residuals');
plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: TV")

# 2) Radio model plots
plt.subplot(3,3,4)
sns.scatterplot(data=train, x=train['TV'], y=residual_tr_radio);
plt.ylabel('residuals');
plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Radio")

plt.subplot(3,3,5)
sns.scatterplot(data=train, x=train['radio'], y=residual_tr_radio);
plt.ylabel('residuals');
plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Radio")

plt.subplot(3,3,6)
sns.scatterplot(data=train, x=train['newspaper'], y=residual_tr_radio);
plt.ylabel('residuals');
plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Radio")

# 3) Newspaper model plots

plt.subplot(3,3,7)
sns.scatterplot(data=train, x=train['TV'], y=residual_tr_news);
plt.ylabel('residuals');
plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Newspaper")

plt.subplot(3,3,8)
sns.scatterplot(data=train, x=train['radio'], y=residual_tr_news);
plt.ylabel('residuals');
plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Newspaper")

plt.subplot(3,3,9)
sns.scatterplot(data=train, x=train['newspaper'], y=residual_tr_news);
```

```
plt.ylabel('residuals');
plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Newspaper")
```

```
Out[109]:
<Axes: >
Out[109]:
<Axes: xlabel='TV', ylabel='sales'>
Out[109]:
Text(0, 0.5, 'residuals')
Out[109]:
(-15.0, 20.0)
Out[109]:
Text(0.5, 1.0, 'Model: TV')
Out[109]:
<Axes: >
Out[109]:
<Axes: xlabel='radio', ylabel='sales'>
Out[109]:
Text(0, 0.5, 'residuals')
Out[109]:
(-15.0, 20.0)
Out[109]:
Text(0.5, 1.0, 'Model: TV')
Out[109]:
<Axes: >
Out[109]:
<Axes: xlabel='newspaper', ylabel='sales'>
Out[109]:
Text(0, 0.5, 'residuals')
Out[109]:
(-15.0, 20.0)
Out[109]:
Text(0.5, 1.0, 'Model: TV')
Out[109]:
<Axes: >
Out[109]:
<Axes: xlabel='TV', ylabel='sales'>
Out[109]:
Text(0, 0.5, 'residuals')
Out[109]:
(-15.0, 20.0)
Out[109]:
```

```
Text(0.5, 1.0, 'Model: Radio')

Out[109]:

<Axes: >

Out[109]:

<Axes: xlabel='radio', ylabel='sales'>

Out[109]:

Text(0, 0.5, 'residuals')

Out[109]:

(-15.0, 20.0)

Out[109]:

Text(0.5, 1.0, 'Model: Radio')

Out[109]:

<Axes: >

Out[109]:

<Axes: xlabel='newspaper', ylabel='sales'>

Out[109]:

Text(0, 0.5, 'residuals')

Out[109]:

(-15.0, 20.0)

Out[109]:

Text(0.5, 1.0, 'Model: Radio')

Out[109]:

<Axes: >

Out[109]:

<Axes: xlabel='TV', ylabel='sales'>

Out[109]:

Text(0, 0.5, 'residuals')

Out[109]:

(-15.0, 20.0)

Out[109]:

Text(0.5, 1.0, 'Model: Newspaper')

Out[109]:

<Axes: >

Out[109]:

<Axes: xlabel='radio', ylabel='sales'>

Out[109]:

Text(0, 0.5, 'residuals')

Out[109]:

(-15.0, 20.0)

Out[109]:

Text(0.5, 1.0, 'Model: Newspaper')

Out[109]:
```

```
<Axes: >
```

Out[109]:

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

Out[109]:

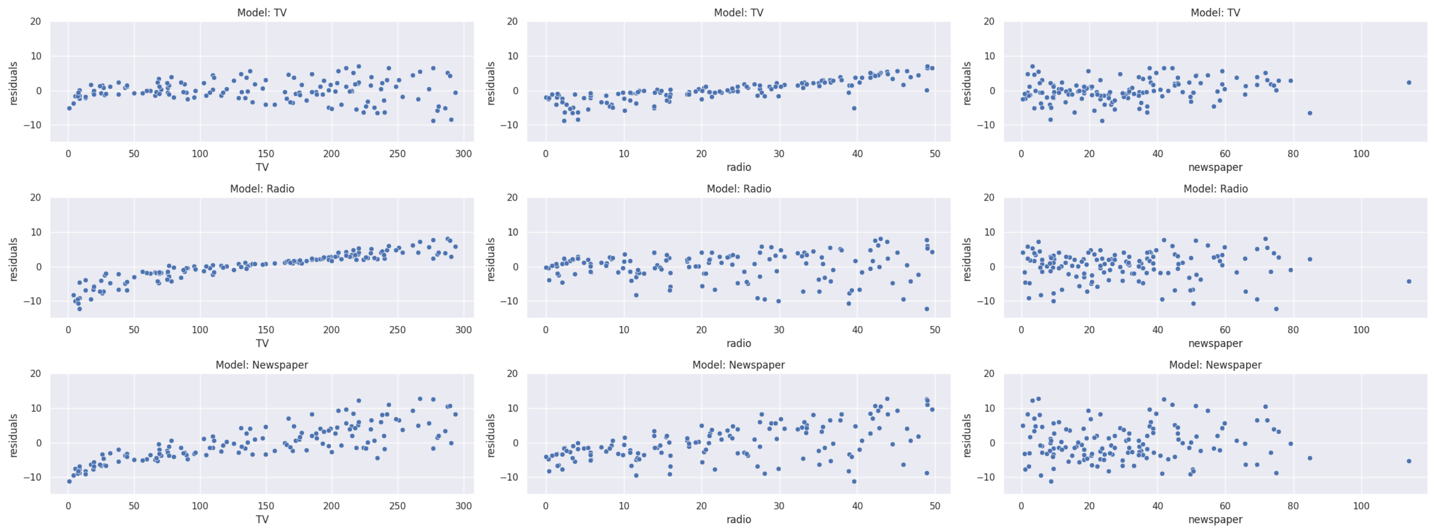
```
Text(0, 0.5, 'residuals')
```

Out[109]:

```
(-15.0, 20.0)
```

Out[109]:

```
Text(0.5, 1.0, 'Model: Newspaper')
```



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

In [110]:

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

Look at coefficients

In [111]:

```
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

In [112]:

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

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

In [113]:

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

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

Comment:

The coefficients are indeed different when comparing the multiple regression model to the three simple linear regressions we ran before. This is expected, because now we are trying to explain sales using three variables simultaneously. In fact, there shouldn't be a direct comparison between (for example) the coefficient of radio ads from the multiple regression vs the coefficient of radio ads from the single regression model. They do not need to be the same, nor they should. The coefficients from the single regressions reveal broadly how much do sales increase when the ads are increased in any three features. However, the multi regression model reveals how much do sales increase given an increase of ads in one feature (say, TV), while controlling for the ads of other features (radio and newspaper).

From the multiple regression model we can observe that the largest coefficient is the one paired with the feature of radio ads, followed by tv, while the coefficient for newspaper is quite small.

In [114]:

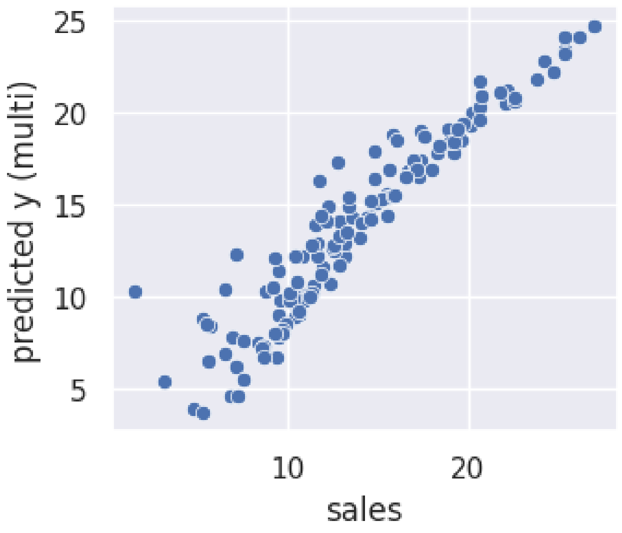
```
train[['TV', 'radio', 'newspaper']].corr()
```

Out[114]:

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

In [115]:

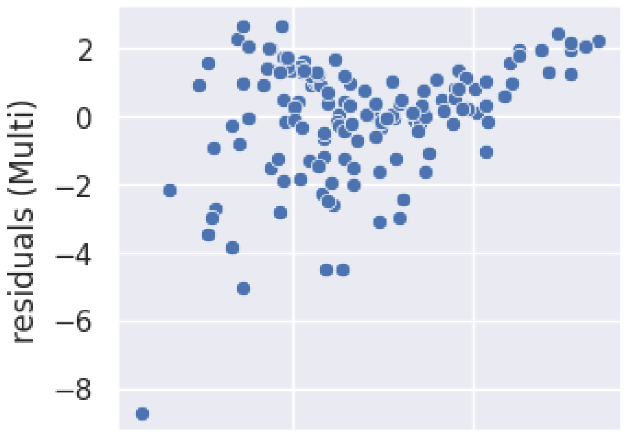
```
# Plot 1: scatter of predicted sales vs actual sales

fig = plt.figure(figsize=(12,3))

plt.subplot(1,3,1)
sns.scatterplot(data=train, x=train["sales"], y=y_pred_tr_multi);
plt.ylabel('predicted y (multi)');
# plt.ylim(0, 25)
```



Comment: we can see that this model gets closer to what we would desire for this plot to look like: a diagonal between the predicted sales and the actual sales.

In [116]:

```
# Plot 2: scatter plot of the residuals vs actual sales

residual_tr_multi = train['sales'] - y_pred_tr_multi

fig = plt.figure(figsize=(12,3))

plt.subplot(1,3,1)
sns.scatterplot(data=train, x=train['sales'], y=residual_tr_multi);
plt.ylabel('residuals (Multi)');
# plt.ylim(-15, 15);
# plt.tight_layout();
```



Comment: from Plot 2 we can't identify a pattern easily. We do see some kind of curved relation between sales and residuals, but it seems to be a weak relation. Compared to the simple regression models, we have a much more randomized relation between the residuals and the actual sales.

In [117]:

```
# Plot 3: residuals vs feature each of the feature (three subplots).


fig = plt.figure(figsize=(24,12))

plt.subplot(3,3,1)
sns.scatterplot(data=train, x=train['TV'], y=residual_tr_multi);
plt.ylabel('residuals');
# plt.ylim(-15, 20);
# plt.tight_layout();
plt.title("Model: Multi")

plt.subplot(3,3,2)
sns.scatterplot(data=train, x=train['radio'], y=residual_tr_multi);
plt.ylabel('residuals');
# plt.ylim(-15, 20);
# plt.tight_layout();
plt.title("Model: Multi")

plt.subplot(3,3,3)
sns.scatterplot(data=train, x=train['newspaper'], y=residual_tr_multi);
plt.ylabel('residuals');
# plt.ylim(-15, 20);
# plt.tight_layout();
plt.title("Model: Multi")
```

Out[117]:

```
<Axes: >
```

Out[117]:

```
<Axes: xlabel='TV', ylabel='sales'>
```

Out[117]:

```
Text(0, 0.5, 'residuals')
```

Out[117]:

```
Text(0.5, 1.0, 'Model: Multi')
```

Out[117]:

```
<Axes: >
```

Out[117]:

```
<Axes: xlabel='radio', ylabel='sales'>
```

Out[117]:

```
Text(0, 0.5, 'residuals')
```

Out[117]:

```
Text(0.5, 1.0, 'Model: Multi')
```

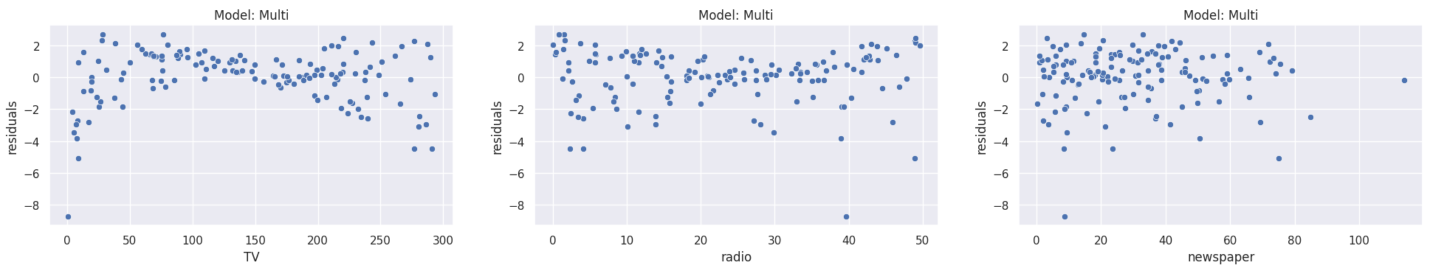
Out[117]:

```
<Axes: >
```

Out[117]:

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

```
Out[117]:
Text(0, 0.5, 'residuals')
Out[117]:
Text(0.5, 1.0, 'Model: Multi')
```



Comment: these plots give very interesting insights. First, the newspaper ads feature seems to be sufficiently uncorrelated with the residuals, meaning that it was properly incorporated in the model. However, the TV ads feature shows a possible non linearity between TV ads and the residuals. This suggests that, for example, we might apply a linear basis transformation to this feature and add it to the model (for example (TV_ads)^2). However, we also see this shape for the radio ads plot vs the residuals. The residuals are somewhat smaller near the center values of TV ads and radio ads. This hints that we might want to try an interaction term between both features.

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.

In [118]:

```
# 1) Adding interactive variables to our datasets
```

```
train = train.assign( newspaper_radio = train['newspaper'] * train['radio'])
test = test.assign( newspaper_radio = test['newspaper'] * test['radio'])

train = train.assign( TV_radio = train['TV'] * train['radio'])
test = test.assign( TV_radio = test['TV'] * test['radio'])

train = train.assign( newspaper_TV = train['newspaper'] * train['TV'])
test = test.assign( newspaper_TV = test['newspaper'] * test['TV'])

train = train.assign( newspaper_radio_TV = train['newspaper'] * train['radio'] * train['
TV'])
test = test.assign( newspaper_radio_TV = test['newspaper'] * test['radio'] * test['TV'])

# train.head()
# test.head()
```

In [119]:

```
# 2) We fit the model

reg_multi_interact = LinearRegression().fit(train[['TV', 'radio', 'newspaper', 'newspape
r_radio', 'TV_radio', 'newspaper_TV', 'newspaper_radio_TV']], train['sales'])

print("Coefficients (TV, radio, newspaper, newspaper_radio, TV_radio, newspaper_TV, newsp
aper_radio_TV):", reg_multi_interact.coef_)
print("Intercept: ", reg_multi_interact.intercept_)
```

```
Coefficients (TV, radio, newspaper, newspaper_radio, TV_radio, newspaper_TV, newspaper_ra
dio_TV): [ 2.07581277e-02  1.19519832e-02  2.03755245e-02 -2.46384818e-05
  1.18859985e-03 -8.06743937e-05 -7.25626214e-07]
Intercept:  6.378007972477893
```

In [120]:

```
# 3) Compute R2 and MSE for training data

y_pred_tr_multi_interact = reg_multi_interact.predict(train[['TV', 'radio', 'newspaper',
'newspaper_radio', 'TV_radio', 'newspaper_TV', 'newspaper_radio_TV']])

r2_tr_multi_interact  = metrics.r2_score(train['sales'], y_pred_tr_multi_interact)
mse_tr_multi_interact = metrics.mean_squared_error(train['sales'], y_pred_tr_multi_inter
act)

print("Multiple interactive terms regression R2:  ", r2_tr_multi_interact)
print("Multiple interactive terms regression MSE: ", mse_tr_multi_interact)
```

```
Multiple interactive terms regression R2:   0.9639737928022052
Multiple interactive terms regression MSE:  0.997910205484344
```

In [121]:

```
# 3.2) Compute R2 and MSE for test data

y_pred_ts_multi_interact = reg_multi_interact.predict(test[['TV', 'radio', 'newspaper',
'newspaper_radio', 'TV_radio', 'newspaper_TV', 'newspaper_radio_TV']])

r2_ts_multi_interact  = metrics.r2_score(test['sales'], y_pred_ts_multi_interact)
mse_ts_multi_interact = metrics.mean_squared_error(test['sales'], y_pred_ts_multi_interac
t)

print("Multiple interactive terms regression R2:  ", r2_ts_multi_interact)
print("Multiple interactive terms regression MSE: ", mse_ts_multi_interact)
```

```
Multiple interactive terms regression R2:   0.978290346306146
Multiple interactive terms regression MSE:  0.5487825962280087
```

In [122]:

```
# 4) Analyze the model performance: plots and residual analysis

# Plot 1: scatter of predicted sales vs actual sales
```

```
fig = plt.figure(figsize=(12,3))

plt.subplot(1,3,1)
sns.scatterplot(data=train, x=train["sales"], y=y_pred_tr_multi_interact);
plt.ylabel('predicted y (multi-interact)');
plt.title("Plot 1: predicted vs actual sales")
# plt.ylim(0, 25)
```

Out[122]:

```
<Axes: >
```

Out[122]:

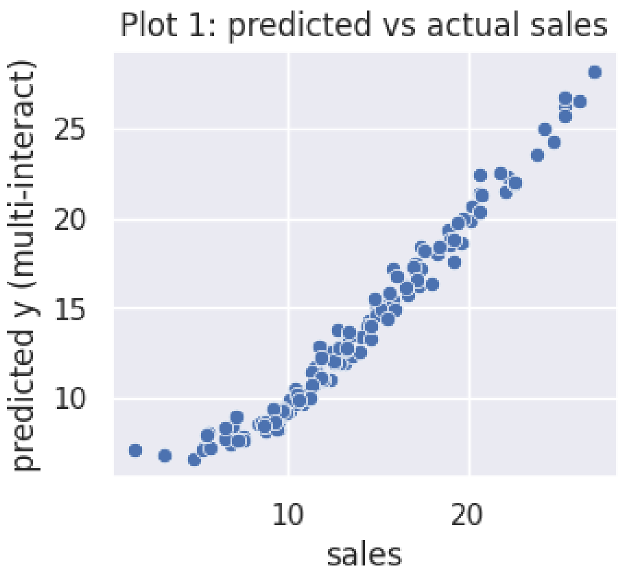
```
<Axes: xlabel='sales'>
```

Out[122]:

```
Text(0, 0.5, 'predicted y (multi-interact)')
```

Out[122]:

```
Text(0.5, 1.0, 'Plot 1: predicted vs actual sales')
```



In [123]:

```
# Plot 2: scatter plot of the residuals vs actual sales

residual_tr_multi_interact = train['sales'] - y_pred_tr_multi_interact

fig = plt.figure(figsize=(12,3))

plt.subplot(1,3,1)
sns.scatterplot(data=train, x=train['sales'], y=residual_tr_multi_interact);
plt.ylabel('residuals (Multi-Interact)');
plt.title("Plot 2: residuals vs actual sales")
# plt.ylim(-15, 15);
# plt.tight_layout();
```

Out[123]:

```
<Axes: >
```

Out[123]:

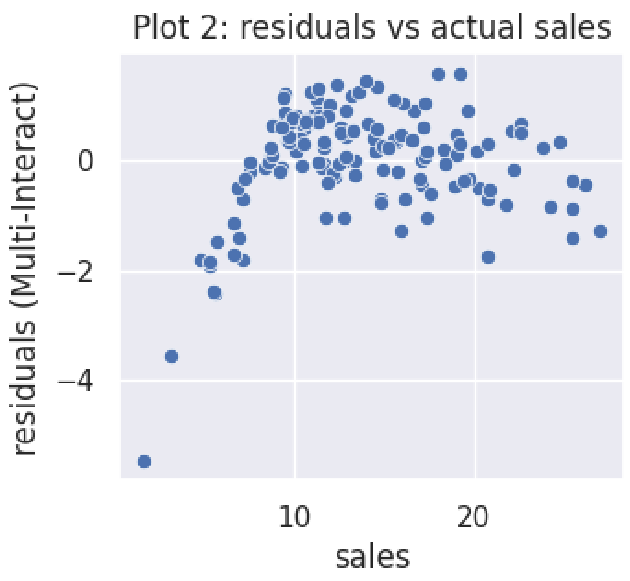
```
<Axes: xlabel='sales', ylabel='sales'>
```

Out[123]:

```
Text(0, 0.5, 'residuals (Multi-Interact)')
```

Out[123]:

```
Text(0.5, 1.0, 'Plot 2: residuals vs actual sales')
```



In [124]:

```
# Plot 3: residuals vs feature each of the feature (three subplots).


fig = plt.figure(figsize=(24,9))
fig.suptitle("Plot 3: residuals vs each incorporated feature", fontsize=14)

plt.subplot(2,4,1)
sns.scatterplot(data=train, x=train['TV'], y=residual_tr_multi_interact);
plt.ylabel('residuals');
# plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Multi-Interact")

plt.subplot(2,4,2)
sns.scatterplot(data=train, x=train['radio'], y=residual_tr_multi_interact);
plt.ylabel('residuals');
# plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Multi-Interact")

plt.subplot(2,4,3)
sns.scatterplot(data=train, x=train['newspaper'], y=residual_tr_multi_interact);
plt.ylabel('residuals');
# plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Multi-Interact")

plt.subplot(2,4,5)
sns.scatterplot(data=train, x=train['newspaper_radio'], y=residual_tr_multi_interact);
plt.ylabel('residuals');
# plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Multi-Interact")

plt.subplot(2,4,6)
sns.scatterplot(data=train, x=train['TV_radio'], y=residual_tr_multi_interact);
plt.ylabel('residuals');
# plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Multi-Interact")

plt.subplot(2,4,7)
sns.scatterplot(data=train, x=train['newspaper_TV'], y=residual_tr_multi_interact);
plt.ylabel('residuals');
# plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Multi-Interact")

plt.subplot(2,4,8)
```

```
sns.scatterplot(data=train, x=train['newspaper_radio_TV'], y=residual_tr_multi_interact)
;
plt.ylabel('residuals');
# plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Multi-Interact")
```

Out[124]:

```
Text(0.5, 0.98, 'Plot 3: residuals vs each incorporated feature')
```

Out[124]:

```
<Axes: >
```

Out[124]:

```
<Axes: xlabel='TV', ylabel='sales'>
```

Out[124]:

```
Text(0, 0.5, 'residuals')
```

Out[124]:

```
Text(0.5, 1.0, 'Model: Multi-Interact')
```

Out[124]:

```
<Axes: >
```

Out[124]:

```
<Axes: xlabel='radio', ylabel='sales'>
```

Out[124]:

```
Text(0, 0.5, 'residuals')
```

Out[124]:

```
Text(0.5, 1.0, 'Model: Multi-Interact')
```

Out[124]:

```
<Axes: >
```

Out[124]:

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

Out[124]:

```
Text(0, 0.5, 'residuals')
```

Out[124]:

```
Text(0.5, 1.0, 'Model: Multi-Interact')
```

Out[124]:

```
<Axes: >
```

Out[124]:

```
<Axes: xlabel='newspaper_radio', ylabel='sales'>
```

Out[124]:

```
Text(0, 0.5, 'residuals')
```

Out[124]:

```
Text(0.5, 1.0, 'Model: Multi-Interact')
```

Out[124]:

```
<Axes: >
```

Out[124]:

```
<Axes: xlabel='TV_radio', ylabel='sales'>
```

Out[124]:

```
Text(0, 0.5, 'residuals')
```

Out[124]:

```
Text(0.5, 1.0, 'Model: Multi-Interact')
```

Out[124]:

```
<Axes: >
```

Out[124]:

```
<Axes: xlabel='newspaper_TV', ylabel='sales'>
```

Out[124]:

```
Text(0, 0.5, 'residuals')
```

Out[124]:

```
Text(0.5, 1.0, 'Model: Multi-Interact')
```

Out[124]:

```
<Axes: >
```

Out[124]:

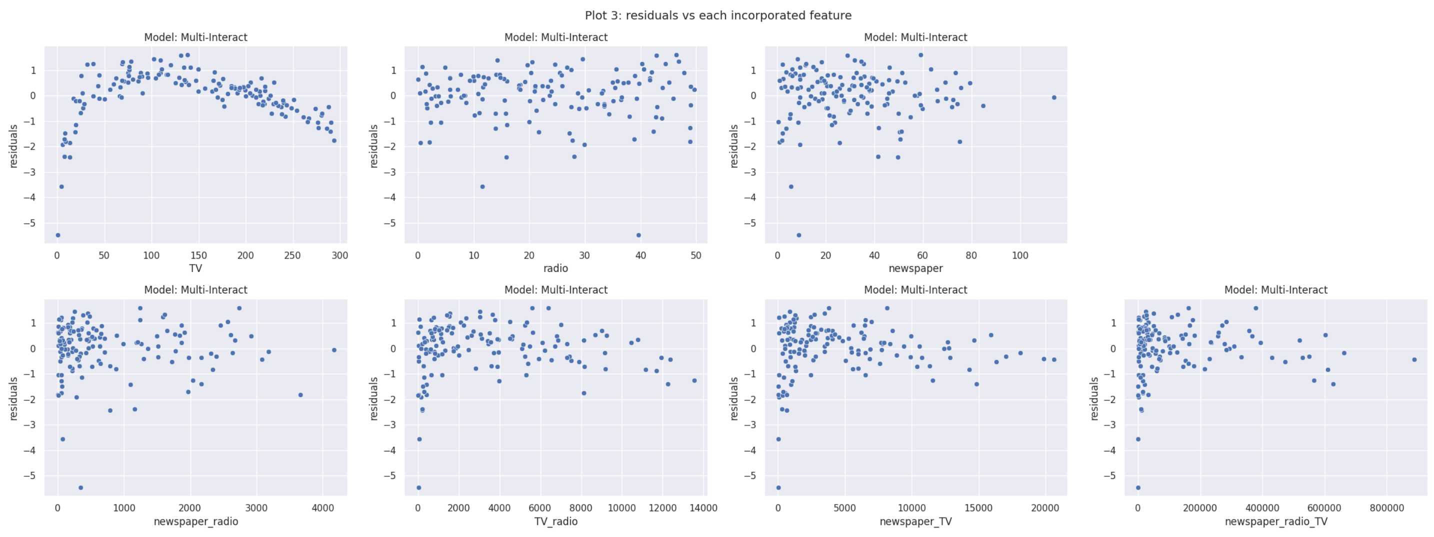
```
<Axes: xlabel='newspaper_radio_TV', ylabel='sales'>
```

Out[124]:

```
Text(0, 0.5, 'residuals')
```

Out[124]:

```
Text(0.5, 1.0, 'Model: Multi-Interact')
```



Comments:

1) In general terms, the model that incorporates interactive terms performs significantly better than both the single regression and multi regression models. The R2 for the model is 0.978, which is very high and close to 1. This supports the assumption that ads are generally employed together simultaneously across different platforms.

2) We can learn about additional opportunities for improval if we take a look at the plots. From Plot 1 we detect a possible non linearity, since the line is slightly curved. From Plot 2 the issue becomes more clear, since the residuals display a downward curve. Plot 3 for residuals vs 'TV' feature apparently reveal the reason behind this. The residuals, when plotted against TV ads, seem to behave in a clear non linear way. It is clear that the model could take advantage of incorporating a linear basis transformation for the feature TV-ads, of the sorts of a polynomial, for example.

We actually implement one last model, incorporating an additional feature of TV ads squared ("TV2"), and the model performance increases significantly. If we check the following lines, we can see that Plot 1 displays

almost a perfect diagonal between predicted sales and actual sales. We can see that Plot 2 shows residuals that behave quite randomly. And finally, Plot 3 shows that the non linear behavior of the residuals when plotted agaisnt the TV ads feature is gone when we incorporate TV^2. Additionally, the R2 of the model also increased. Therefore, this was a correct approach towards improving the model.

In [124]:

```

```

In [125]:

```
# 1) Adding TV^2 as a variable to our dataset
train = train.assign( TV2 = train['TV'] * train['TV'])
test = test.assign( TV2 = test['TV'] * test['TV'])

# 2) We fit the model

reg_multi_interact_tv2 = LinearRegression().fit(train[['TV', 'radio', 'newspaper', 'news
paper_radio', 'TV_radio', 'newspaper_TV', 'newspaper_radio_TV', 'TV2']], train['sales'])

print("Coefficients (TV, radio, newspaper, newspaper_radio, TV_radio, newspaper_TV, newsp
aper_radio_TV, TV2):", reg_multi_interact_tv2.coef_)
print("Intercept: ", reg_multi_interact_tv2.intercept_)
```

```
Coefficients (TV, radio, newspaper, newspaper_radio, TV_radio, newspaper_TV, newspaper_ra
dio_TV, TV2): [ 5.58662819e-02  2.15105054e-02  1.48547042e-02 -1.64336343e-05
  1.11409003e-03 -1.02296202e-04  9.78191649e-07 -1.20642479e-04]
Intercept:  4.7936436970179646
```

In [126]:

```
# 3) Compute R2 and MSE for training data

y_pred_tr_multi_interact_tv2 = reg_multi_interact_tv2.predict(train[['TV', 'radio', 'new
spaper', 'newspaper_radio', 'TV_radio', 'newspaper_TV', 'newspaper_radio_TV', 'TV2']])

r2_tr_multi_interact  = metrics.r2_score(train['sales'], y_pred_tr_multi_interact_tv2)
mse_tr_multi_interact = metrics.mean_squared_error(train['sales'], y_pred_tr_multi_inter
act_tv2)

print("Multiple interactive terms + TV2 regression R2:  ", r2_tr_multi_interact)
print("Multiple interactive terms + TV2 regression MSE: ", mse_tr_multi_interact)
```

```
Multiple interactive terms + TV2 regression R2:   0.9849867419704118
Multiple interactive terms + TV2 regression MSE:  0.4158606906089431
```

In [127]:

```
# 3.2) Compute R2 and MSE for test data

y_pred_ts_multi_interact_tv2 = reg_multi_interact_tv2.predict(test[['TV', 'radio', 'news
paper', 'newspaper_radio', 'TV_radio', 'newspaper_TV', 'newspaper_radio_TV', 'TV2']])

r2_ts_multi_interact  = metrics.r2_score(test['sales'], y_pred_ts_multi_interact_tv2)
mse_ts_multi_interact = metrics.mean_squared_error(test['sales'], y_pred_ts_multi_interac
t_tv2)

print("Multiple interactive terms + TV2 regression R2:  ", r2_ts_multi_interact)
print("Multiple interactive terms + TV2 regression MSE: ", mse_ts_multi_interact)
```

```
Multiple interactive terms + TV2 regression R2:   0.98874542148482
Multiple interactive terms + TV2 regression MSE:  0.28449633071581154
```

In [128]:

```
# 4) Analyze the model performance: plots and residual analysis

# Plot 1: scatter of predicted sales vs actual sales

fig = plt.figure(figsize=(12,3))
```

```
plt.subplot(1,3,1)
sns.scatterplot(data=train, x=train["sales"], y=y_pred_tr_multi_interact_tv2);
plt.ylabel('predicted y (multi-interac-tv2)');
plt.title("Plot 1: predicted vs actual sales")
# plt.ylim(0, 25)
```

Out[128]:

```
<Axes: >
```

Out[128]:

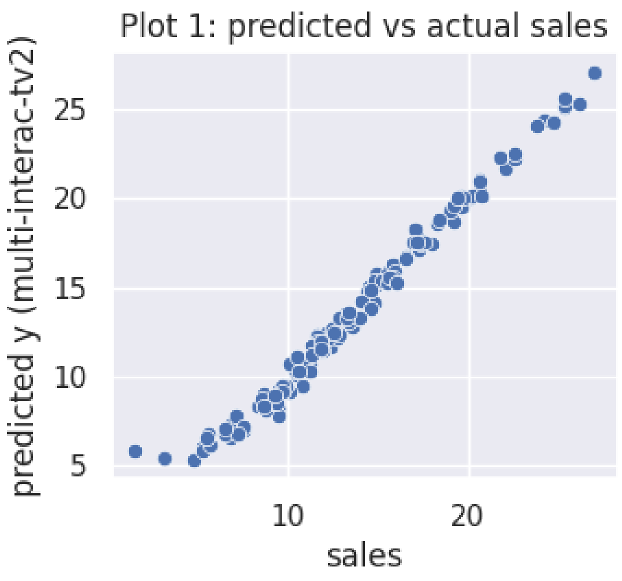
```
<Axes: xlabel='sales'>
```

Out[128]:

```
Text(0, 0.5, 'predicted y (multi-interac-tv2)')
```

Out[128]:

```
Text(0.5, 1.0, 'Plot 1: predicted vs actual sales')
```



In [129]:

```
# Plot 2: scatter plot of the residuals vs actual sales

residual_tr_multi_interact_tv2 = train['sales'] - y_pred_tr_multi_interact_tv2

fig = plt.figure(figsize=(12,3))

plt.subplot(1,3,1)
sns.scatterplot(data=train, x=train['sales'], y=residual_tr_multi_interact_tv2);
plt.ylabel('residuals (Multi-Interact-TV2)');
plt.title("Plot 2: residuals vs actual sales")
# plt.ylim(-15, 15);
# plt.tight_layout();
```

Out[129]:

```
<Axes: >
```

Out[129]:

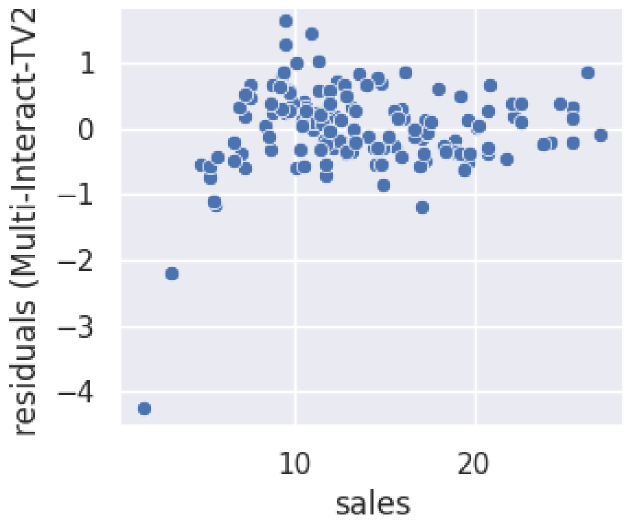
```
<Axes: xlabel='sales', ylabel='sales'>
```

Out[129]:

```
Text(0, 0.5, 'residuals (Multi-Interact-TV2)')
```

Out[129]:

```
Text(0.5, 1.0, 'Plot 2: residuals vs actual sales')
```



In [130]:

```
# Plot 3: residuals vs feature each of the feature (three subplots).


fig = plt.figure(figsize=(24,9))
fig.suptitle("Plot 3: residuals vs each incorporated feature", fontsize=14)

plt.subplot(2,4,1)
sns.scatterplot(data=train, x=train['TV'], y=residual_tr_multi_interact_tv2);
plt.ylabel('residuals');
# plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Multi-Interact")

plt.subplot(2,4,2)
sns.scatterplot(data=train, x=train['TV2'], y=residual_tr_multi_interact_tv2);
plt.ylabel('residuals');
# plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Multi-Interact")

plt.subplot(2,4,3)
sns.scatterplot(data=train, x=train['radio'], y=residual_tr_multi_interact_tv2);
plt.ylabel('residuals');
# plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Multi-Interact")

plt.subplot(2,4,4)
sns.scatterplot(data=train, x=train['newspaper'], y=residual_tr_multi_interact_tv2);
plt.ylabel('residuals');
# plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Multi-Interact")

plt.subplot(2,4,5)
sns.scatterplot(data=train, x=train['newspaper_radio'], y=residual_tr_multi_interact_tv2
);
plt.ylabel('residuals');
# plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Multi-Interact")

plt.subplot(2,4,6)
sns.scatterplot(data=train, x=train['TV_radio'], y=residual_tr_multi_interact_tv2);
plt.ylabel('residuals');
# plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Multi-Interact")

plt.subplot(2,4,7)
sns.scatterplot(data=train, x=train['newspaper_TV'], y=residual_tr_multi_interact_tv2);
```

```
plt.ylabel('residuals');
# plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Multi-Interact")

plt.subplot(2,4,8)
sns.scatterplot(data=train, x=train['newspaper_radio_TV'], y=residual_tr_multi_interact_t
v2);
plt.ylabel('residuals');
# plt.ylim(-15, 20);
plt.tight_layout();
plt.title("Model: Multi-Interact")
```

Out[130]:

```
Text(0.5, 0.98, 'Plot 3: residuals vs each incorporated feature')
```

Out[130]:

```
<Axes: >
```

Out[130]:

```
<Axes: xlabel='TV', ylabel='sales'>
```

Out[130]:

```
Text(0, 0.5, 'residuals')
```

Out[130]:

```
Text(0.5, 1.0, 'Model: Multi-Interact')
```

Out[130]:

```
<Axes: >
```

Out[130]:

```
<Axes: xlabel='TV2', ylabel='sales'>
```

Out[130]:

```
Text(0, 0.5, 'residuals')
```

Out[130]:

```
Text(0.5, 1.0, 'Model: Multi-Interact')
```

Out[130]:

```
<Axes: >
```

Out[130]:

```
<Axes: xlabel='radio', ylabel='sales'>
```

Out[130]:

```
Text(0, 0.5, 'residuals')
```

Out[130]:

```
Text(0.5, 1.0, 'Model: Multi-Interact')
```

Out[130]:

```
<Axes: >
```

Out[130]:

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

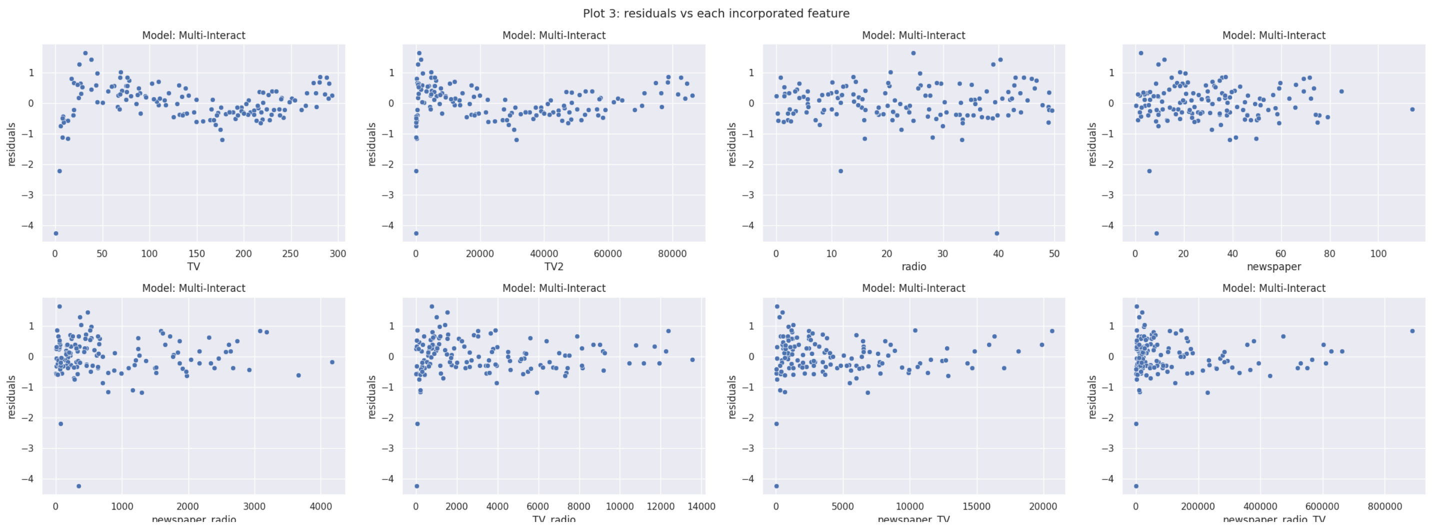
Out[130]:

```
Text(0, 0.5, 'residuals')
```

Out[130]:

```
Text(0.5, 1.0, 'Model: Multi-Interact')
```

```
Out[130]:

<Axes: >

Out[130]:

<Axes: xlabel='newspaper_radio', ylabel='sales'>

Out[130]:

Text(0, 0.5, 'residuals')

Out[130]:

Text(0.5, 1.0, 'Model: Multi-Interact')

Out[130]:

<Axes: >

Out[130]:

<Axes: xlabel='TV_radio', ylabel='sales'>

Out[130]:

Text(0, 0.5, 'residuals')

Out[130]:

Text(0.5, 1.0, 'Model: Multi-Interact')

Out[130]:

<Axes: >

Out[130]:

<Axes: xlabel='newspaper_TV', ylabel='sales'>

Out[130]:

Text(0, 0.5, 'residuals')

Out[130]:

Text(0.5, 1.0, 'Model: Multi-Interact')

Out[130]:

<Axes: >

Out[130]:

<Axes: xlabel='newspaper_radio_TV', ylabel='sales'>

Out[130]:

Text(0, 0.5, 'residuals')

Out[130]:

Text(0.5, 1.0, 'Model: Multi-Interact')
```



newspaper_radio TV_radio newspaper_TV newspaper_radio_TV