Appendix A

Please use this appendix to correlate with the corresponding questions from my homework, thanks.

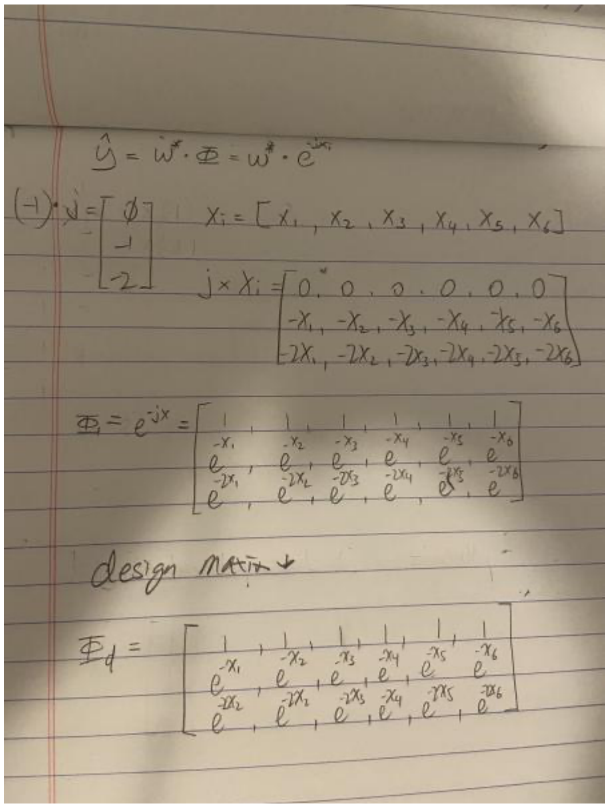
2.3
1.png

$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

2.3
2.png

$$R2 = 1 - \frac{MSE}{\sigma_y^2} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y}_i)^2}$$

3.a



$\hat{y} = w^* \cdot \Phi = w^* \cdot e^{-jx_i}$

$(-1) \cdot j = \begin{bmatrix} \emptyset \\ -1 \\ -2 \end{bmatrix}$ $\quad X_i = [x_1, x_2, x_3, x_4, x_5, x_6]$

$$j \times X_i = \begin{bmatrix} 0, & 0, & 0, & 0, & 0, & 0 \\ -x_1, & -x_2, & -x_3, & -x_4, & -x_5, & -x_6 \\ -2x_1, & -2x_2, & -2x_3, & -2x_4, & -2x_5, & -2x_6 \end{bmatrix}$$

$$\Phi_i = e^{-jx} = \begin{bmatrix} 1, & 1, & 1, & 1, & 1, & 1 \\ e^{-x_1}, & e^{-x_2}, & e^{-x_3}, & e^{-x_4}, & e^{-x_5}, & e^{-x_6} \\ e^{-2x_1}, & e^{-2x_2}, & e^{-2x_3}, & e^{-2x_4}, & e^{-2x_5}, & e^{-2x_6} \end{bmatrix}$$

design matrix ↓

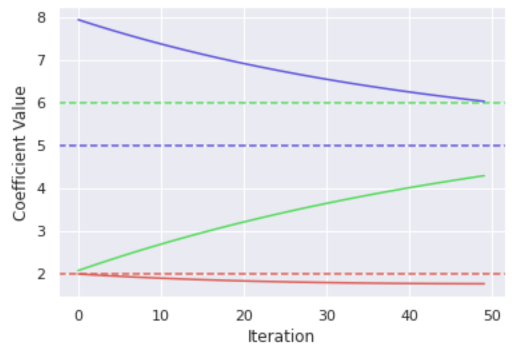
$$\Phi_d = \begin{bmatrix} 1, & 1, & 1, & 1, & 1, & 1 \\ e^{-x_1}, & e^{-x_2}, & e^{-x_3}, & e^{-x_4}, & e^{-x_5}, & e^{-x_6} \\ e^{-2x_2}, & e^{-2x_2}, & e^{-2x_3}, & e^{-x_4}, & e^{-2x_5}, & e^{-2x_6} \end{bmatrix}$$

3.b
Screen Shot 2021-06-27 at 1.39.51 PM.png

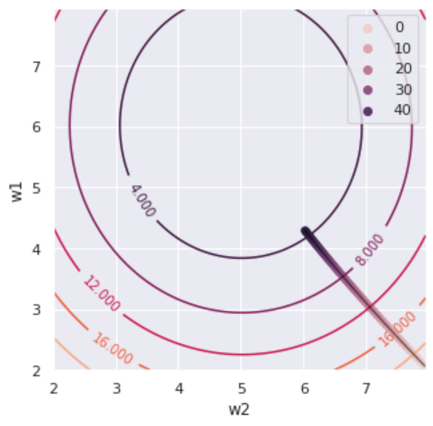
$$\hat{y_i} = \sum_{j=0}^{d} w_j e^{-jx_i}$$

4.1

```
#learning rate: 0.0002
```

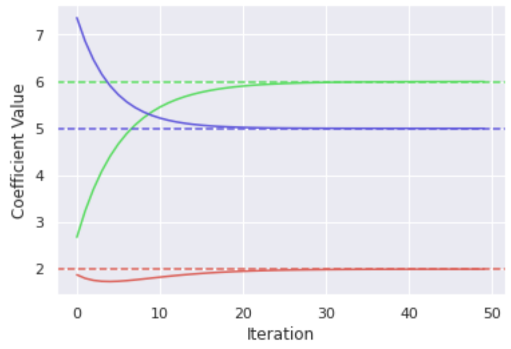


```
plt.figure(figsize=(5,5));
X1, X2 = np.meshgrid(coefs, coefs);
p = plt.contour(X1, X2, mses_coefs, levels=5);
plt.clabel(p, inline=1, fontsize=10);
plt.xlabel('w2');
plt.ylabel('w1');
sns.lineplot(x=w_steps[:,2], y=w_steps[:,1], co
sns.scatterplot(x=w_steps[:,2], y=w_steps[:,1],
```

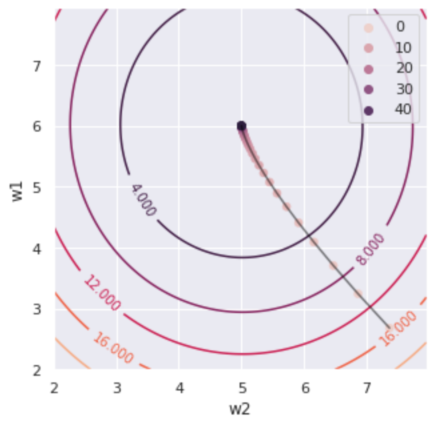


4.2

```
#learning rate: 0.002
```

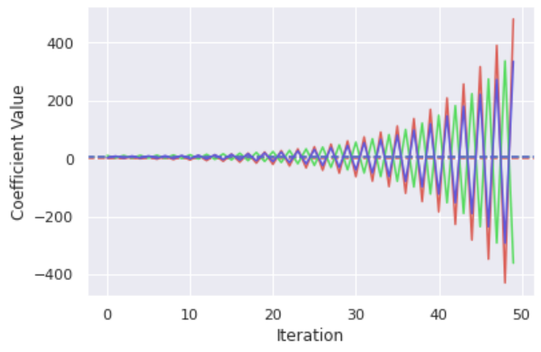


```
plt.figure(figsize=(5,5));
X1, X2 = np.meshgrid(coefs, coefs);
p = plt.contour(X1, X2, mses_coefs, levels=5);
plt.clabel(p, inline=1, fontsize=10);
plt.xlabel('w2');
plt.ylabel('w1');
sns.lineplot(x=w_steps[:,2], y=w_steps[:,1], c
sns.scatterplot(x=w_steps[:,2], y=w_steps[:,1]
```

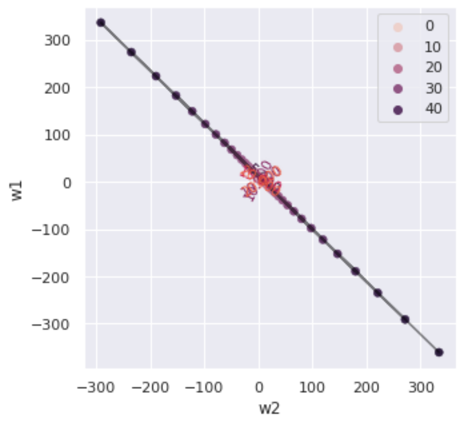


4.3

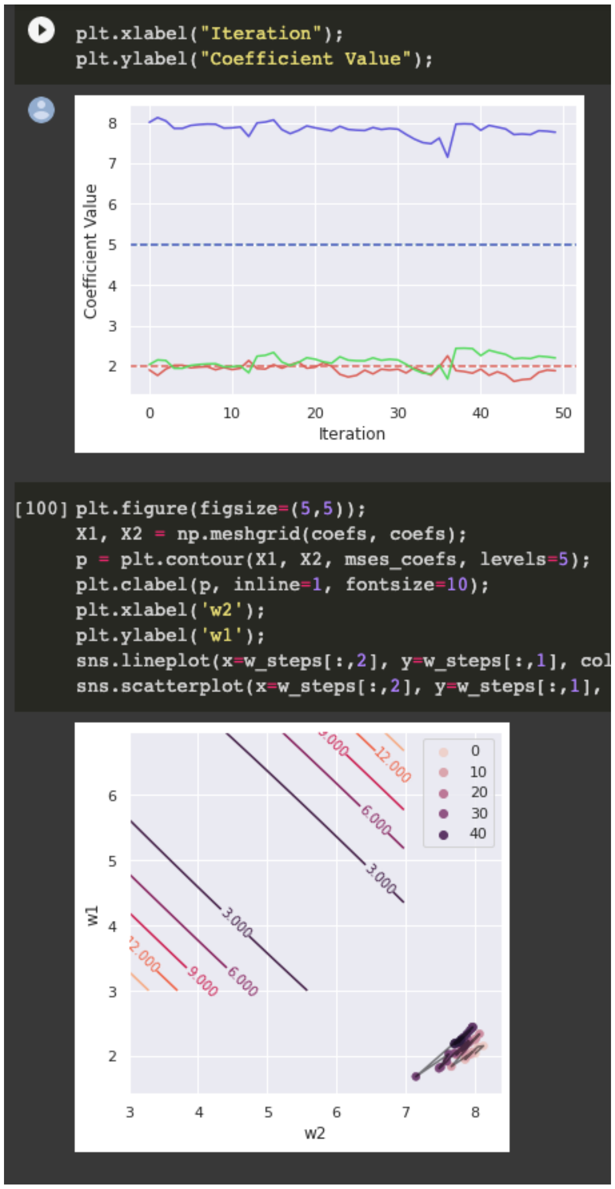
```
#learning rate: 0.02
```



```
plt.figure(figsize=(5,5));
X1, X2 = np.meshgrid(coefs, coefs);
p = plt.contour(X1, X2, mses_coefs, levels=5);
plt.clabel(p, inline=1, fontsize=10);
plt.xlabel('w2');
plt.ylabel('w1');
sns.lineplot(x=w_steps[:,2], y=w_steps[:,1], colo
sns.scatterplot(x=w_steps[:,2], y=w_steps[:,1], h
```



4.4

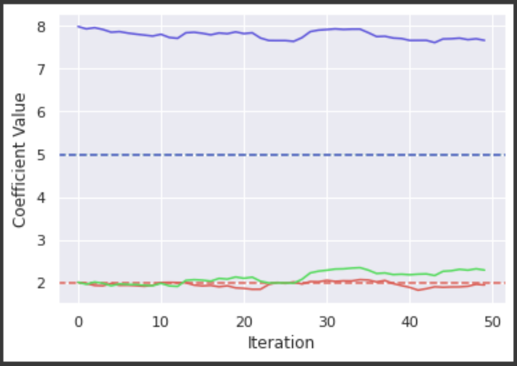


```
plt.xlabel("Iteration");
plt.ylabel("Coefficient Value");
```

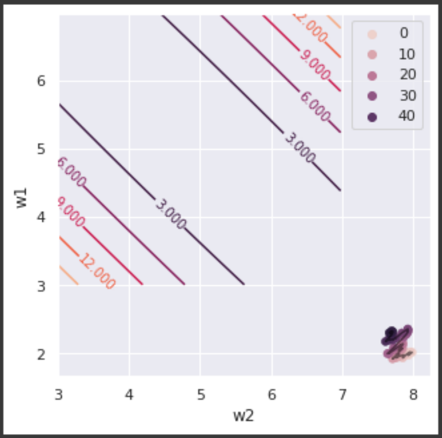
```
[100] plt.figure(figsize=(5,5));
      X1, X2 = np.meshgrid(coefs, coefs);
      p = plt.contour(X1, X2, mses_coefs, levels=5);
      plt.clabel(p, inline=1, fontsize=10);
      plt.xlabel('w2');
      plt.ylabel('w1');
      sns.lineplot(x=w_steps[:,2], y=w_steps[:,1], col
      sns.scatterplot(x=w_steps[:,2], y=w_steps[:,1],
```

4.5

```
plt.xlabel("Iteration");
plt.ylabel("Coefficient Value");
```

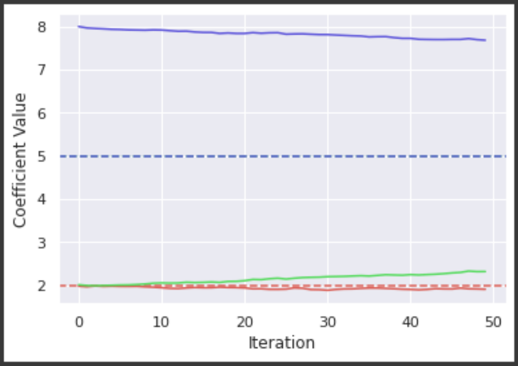


```
[134] plt.figure(figsize=(5,5));
      X1, X2 = np.meshgrid(coefs, coefs);
      p = plt.contour(X1, X2, mses_coefs, levels=5);
      plt.clabel(p, inline=1, fontsize=10);
      plt.xlabel('w2');
      plt.ylabel('w1');
      sns.lineplot(x=w_steps[:,2], y=w_steps[:,1], color
      sns.scatterplot(x=w_steps[:,2], y=w_steps[:,1], hu
```

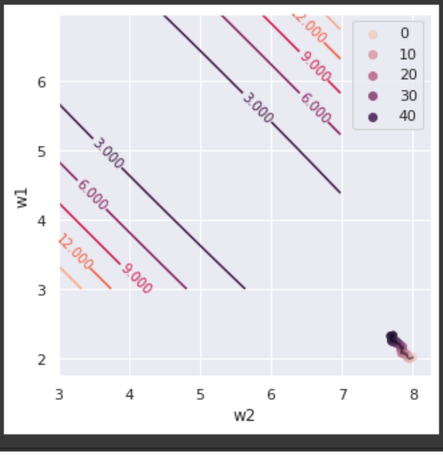


4.6

```
    plt.xlabel("Iteration");
    plt.ylabel("Coefficient Value");
```



```
168] plt.figure(figsize=(5,5));
     X1, X2 = np.meshgrid(coefs, coefs);
     p = plt.contour(X1, X2, mses_coefs, levels=5);
     plt.clabel(p, inline=1, fontsize=10);
     plt.xlabel('w2');
     plt.ylabel('w1');
     sns.lineplot(x=w_steps[:,2], y=w_steps[:,1], color
     sns.scatterplot(x=w_steps[:,2], y=w_steps[:,1], hu
```



ElainaH-2-advertising-hw

June 28, 2021

1 Assignment: Linear regression on the Advertising data

TODO: Edit this cell to fill in your NYU Net ID and your name:

- **Net ID**: N15610800
- **Name**: Elaina(Yaogui) Huang

The following content contains 2 parts. Part I is from the short-answer PDF questions, Part II is from the colab questions.

2 Part I.

1. **Residuals**

False. Residuals can tell the systematic error if it is resulted from wrong implemented regression model. When the majority/all of the residuals are very small to 0, this means the model fits the data pattern, in this sense, the remaining large value residual could be due to the randomness/noises that no model would be able to learn from, or could be due to outlier/high leverage point.

2. **Regression metrics**

- 2.1

(training R2 when N(0, 2>=0)): 0 <= R2 <= 1
When the epsilon is not zero, stochastic error will present in the data, the type of error that the model cannot learn from. In this case, the linear model will have residuals which cannot be fully explained by the model; however, the model will still be able to explain some data points, so the R2 value will not fall below 0, the worst performance is when the linear model is same as the mean. Since the epsilon could equal to 0, in this case, no stochastic error is generated, the R2 can equal to 1.

- 2.2

(testing R2 when N(0, 2>=0)): 0 <= R2 <= 1

Because the test data is also generated through the same function as the training had, the test data should have similar pattern as the training data. Therefore, the R2 value should fall in the same range. Best case scenario is having the test data pattern similar to training data, but is the test data points have different pattern despite both dataset generated from same linear function, the linear model could still perform badly and the mean could be the best prediction for the test dataset.

- 2.3

(training R2 when N(0, 0)): R2 = 1.

Because there are none stochastic error, which meaning the least squares fitted linear regression will have 0 variance in terms of

1.png

Therefore the MSE will be 0, thus R2 = 1.

2.png

- 2.4

(testing R2 when N(0, 0)): R2 = 1.

Because the test data is also generated from the y = A + Bx when N(0, 0); therefore, the test data follows same pattern as the training data, and the linear model will fit just fine.

- 2.5

(training R2 when N(0, 0)): R = 1.

Same reason as described in 2.3. Because there are no stochastic error, no noises, the linear model should fit the linear function generated data very well.

- 2.6

(testing R2 when N(0, 0)): R < 1.

Because the testing data are sampled from a different linear function than the training data. If the test data linear function have very similar slop & intercept, the prediction result might be okay, but if the slope&intercept have great differences, then the model created from training dataset will not be useful to predict on test dataset, therefore the R will be negative.

3. **Linear Basis Function Regression**

- 3.a

Please see Apendix A 3.a, because the picture is too big, significantly slows down my colab, therefore I have to remove it from here.

- 3.b

Screen Shot 2021-06-27 at 1.39.51 PM.png

[2]:
```
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"
```

[3]:
```
# expect the pass-in array to be in feature-label pair as in: [x1,y1],[x2,y2],␣
↪etc.
def DesignMatrix(arr, d):
  feature = np.array(arr[:,0]) # feature = [x1, x2, x3...]

  dimension = np.vstack(np.array([integer*-1 for integer in␣
↪[*range(0,d+1,1)]])) # dimension = [0, -1, -2...]

  # y_hat = optimal_w * phi
  # phi = e^(-j*x)
  jx_exponent = feature * dimension
  constant_e = np.exp(1)

  X = np.array([(constant_e ** integer) for integer in jx_exponent]) #the phi
  X = np.hstack((np.ones(X.shape[0])[:,None], X))

  return X


x = np.array([[1,2],[3,4],[5,6],[7,8],[9,10],[11,12]])
d = 3
DesignMatrix(x, d)
```

[3]:
```
array([[1.00000000e+00, 1.00000000e+00, 1.00000000e+00, 1.00000000e+00,
        1.00000000e+00, 1.00000000e+00, 1.00000000e+00],
       [1.00000000e+00, 3.67879441e-01, 4.97870684e-02, 6.73794700e-03,
        9.11881966e-04, 1.23409804e-04, 1.67017008e-05],
       [1.00000000e+00, 1.35335283e-01, 2.47875218e-03, 4.53999298e-05,
        8.31528719e-07, 1.52299797e-08, 2.78946809e-10],
       [1.00000000e+00, 4.97870684e-02, 1.23409804e-04, 3.05902321e-07,
        7.58256043e-10, 1.87952882e-12, 4.65888615e-15]])
```

4. **Gradient Descent**

- 4.a.i & 4.a.ii & 4.a.iii

When learning rate is 0.0002, the estimated w is (4.29222615, 6.17249588)

3 coefficients appear to converge within 50 iterations. Though not all coefficients have reached the true value, it's getting very close.

Plese see screenshot on Apendix A 4.1

When learning rate is 0.002, the estimated w is (4.29222615, 6.17249588)

3 coefficients have converged to true value before finishing the 50 iterations.

Plese see screenshot on Apendix A 4.2

When learning rate is 0.02, the estimated w is (-360.77897346, 334.83484636)

3 coefficients overshoot the true value, when the 50 iteration is done, none of the coefficients is near the true value.

Plese see screenshot on Apendix A 4.3

- 4.b.i & 4.b.ii

Lr = 0.1, n=1

Very bumpy, only one of the coefficients is coverged. No visual on green's true value line, blue is very bumpy and very difficult to show if it is converging or diverging.

Plese see screenshot on Apendix A 4.4

Lr = 0.01, n=10

Only one of the coefficients is coverged. No visual on green's true value line, howeve, the blue line seems to gradually but slowly converging.

Plese see screenshot on Apendix A 4.5

Lr = 0.001, n=100

Lines are smoothier. Only red line is coverged, green's true value line is not visible, and blue line seems to converging in a very slow pace.

Plese see screenshot on Apendix A 4.6

3 Part II.

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.

3.0.1 1. 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. However, you should add a text cell at the end of this section, in which you write your comments and observations.

Read in data

[9]:
```
url = 'https://www.statlearning.com/s/Advertising.csv'
df  = pd.read_csv(url, index_col=0)
df.head()
```

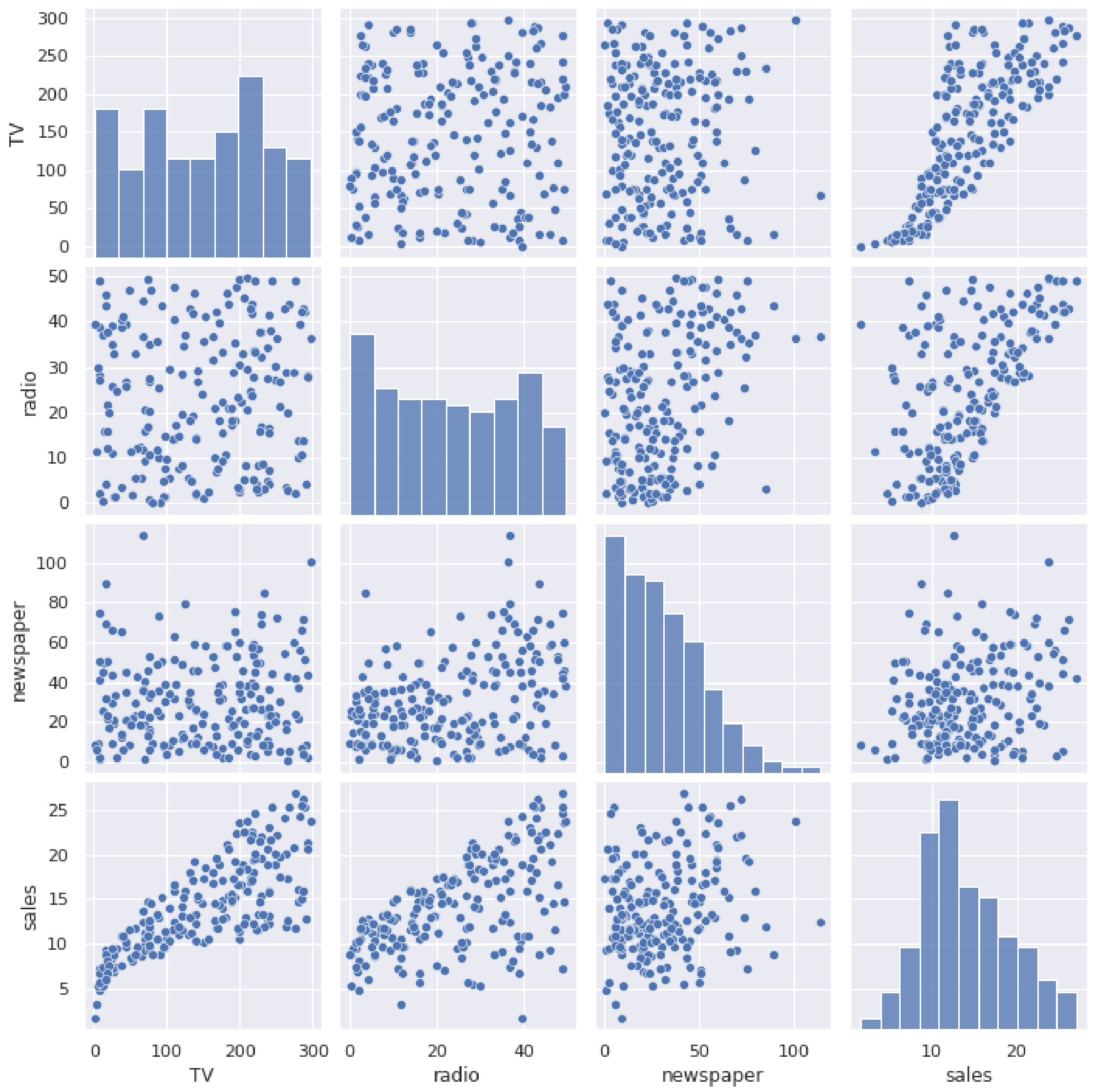
[9]:
```
      TV  radio  newspaper  sales
1  230.1   37.8       69.2   22.1
2   44.5   39.3       45.1   10.4
3   17.2   45.9       69.3    9.3
4  151.5   41.3       58.5   18.5
5  180.8   10.8       58.4   12.9
```

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

[10]:
```
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.

Observations:

- When the sales around 15 thousands dollars, the newspaper appear to have the best saling performance, as the sale increase, the newspaper does not appear to increase as much as when it had the sales around 15 thousands dollars.
- The radio is mostly balanced, based on the plot of radio-radio, we can clearly see the second peak is when radio reach around 40 units.
- The TV has several high-leverage sales and the third peak of TV sales in when TV has around 210 units, after that the sales decrease.
- Based on the sale barplot, the highest sale appear to be among teens.

Split up data We will use 70% of the data for training and the remaining 30% to test the regression model.

[11]:
```
train, test = train_test_split(df, test_size=0.3)
```

[12]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 140 entries, 73 to 78
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
```

[13]:
```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 60 entries, 50 to 51
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
```

3.0.2 2. 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 ads seems to have the greatest association with increased product sales? Which regression model is most effective at predicting product sales?

The code in this section is provided for you. However, you should add text cells in which you write your comments, observations, and answers to the questions.

Fit a simple linear regression

[14]:
```
reg_tv    = LinearRegression().fit(train[['TV']], train['sales'])
reg_radio = LinearRegression().fit(train[['radio']], train['sales'])
reg_news  = LinearRegression().fit(train[['newspaper']], train['sales'])
```

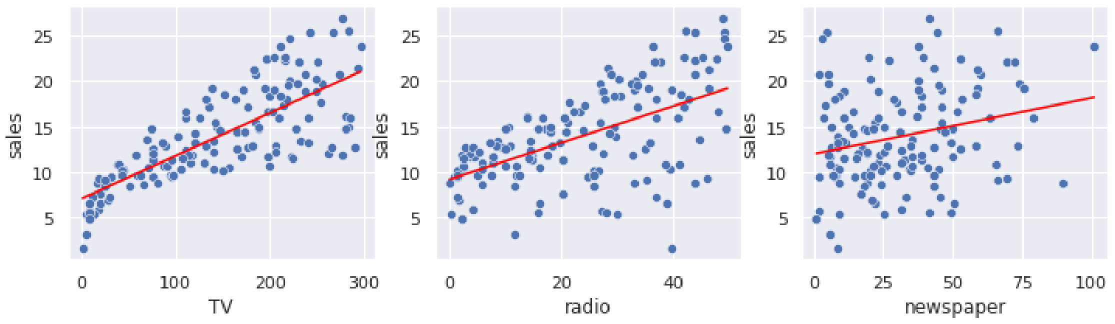
Look at coefficients

[15]:
```
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        :  0.04719665040996279 7.062289992833867
Radio     :  0.2022800349183064 9.169211387176652
Newspaper:  0.0619109119187026 11.977236369032916
```

Plot data and regression line

[16]:
```
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 **------- Section Break ---------**

In this section break, I created a model based on the dataset to see radio & sales with a train-test-split in 50-50 percent.

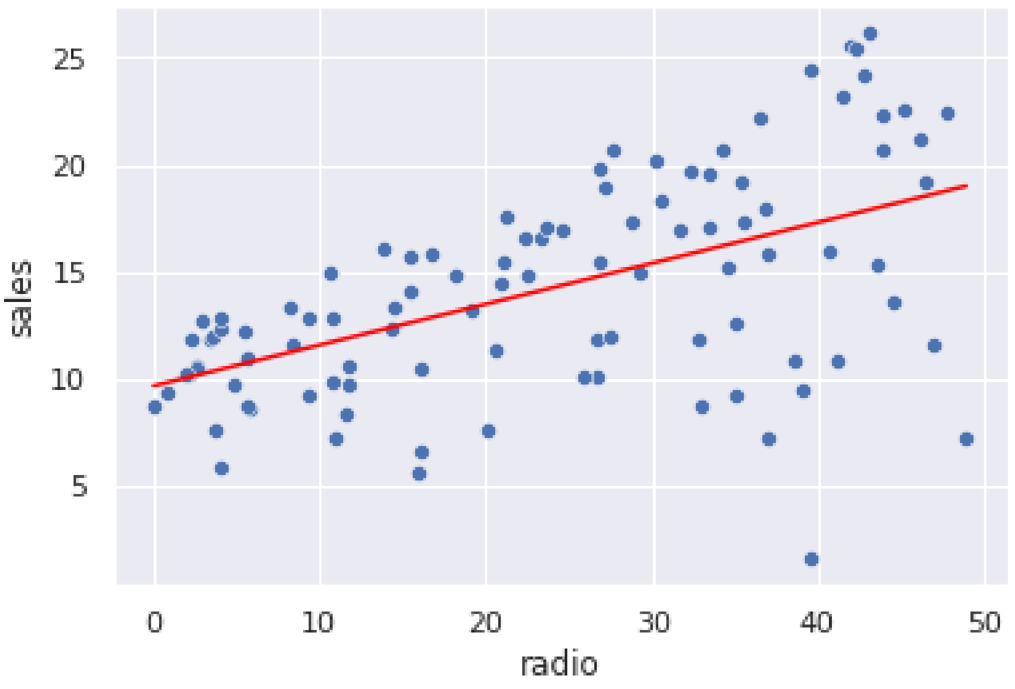
[17]:
```
train2, test2 = train_test_split(df, test_size=0.5)
```

[18]:
```
reg2_radio = LinearRegression().fit(train2[['radio']], train2['sales'])
```

[19]:
```
print("Radio  :", reg2_radio.coef_[0], reg2_radio.intercept_)
```

```
Radio  : 0.19077692354545595 9.678540796590418
```

[20]:
```
sns.scatterplot(data=train2, x="radio", y="sales");
sns.lineplot(data=train2, x="radio", y=reg2_radio.predict(train2[['radio']]),␣
 ↪color='red');
```



[21]:
```
y_pred_tr_radio2 = reg2_radio.predict(train2[['radio']])
r2_tr_radio2 = metrics.r2_score(train2['sales'], y_pred_tr_radio2)
print("Radio train R2   : ", r2_tr_radio2)
y_pred_ts_radio2 = reg2_radio.predict(test2[['radio']])
r2_ts_radio2 = metrics.r2_score(test2['sales'], y_pred_ts_radio2   )
print("Radio  test R2 : ", r2_ts_radio2)

mse_tr_radio2 = metrics.mean_squared_error(train2['sales'], y_pred_tr_radio2)
print("Radio train R2   : ", mse_tr_radio2)
mse_ts_radio2 = metrics.mean_squared_error(test2['sales'], y_pred_ts_radio2)
print("Radio  test R2 : ", mse_ts_radio2)
```

```
Radio train R2   :  0.2947052207775023
Radio  test R2 :  0.36182783054973544
Radio train R2   :  18.197380422902803
Radio  test R2 :  18.065431379260325
```

------------ Section Break -----------

[22]:
```
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']])
```

[23]:
```
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       :  0.6028869509124667
Radio    :  0.32336492488721547
Newspaper:  0.06214993721258777
```

[24]:
```
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       :  10.78275774100348
Radio    :  18.372582091601412
Newspaper:  25.465317867689873
```

[25]:
```
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']])
```

[26]:
```
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       :  0.6306336528754073
Radio    :  0.34577918862035617
Newspaper:  0.016616461330800525
```

[27]:
```python
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       :  9.886607393675698
Radio    :  17.511135925712097
Newspaper:  26.32163714943526
```

[28]:
```python
# To better visualize the scores, here I generate a table to compare
data = {
    'R2':[r2_tr_tv, r2_tr_radio, r2_tr_news, r2_ts_tv, r2_ts_radio, r2_ts_news␣
 ↪],
    'MSE':[mse_tr_tv, mse_tr_radio, mse_tr_news, mse_ts_tv, mse_ts_radio,␣
 ↪mse_ts_news]
}

dff = pd.DataFrame(data, index = ['tv_training', 'radio_training',␣
 ↪'news_training', 'tv_test', 'radio_test', 'news_test'])

dff
```

[28]:
```
                      R2        MSE
tv_training     0.602887  10.782758
radio_training  0.323365  18.372582
news_training   0.062150  25.465318
tv_test         0.630634   9.886607
radio_test      0.345779  17.511136
news_test       0.016616  26.321637
```

Observations:

Overall, they do not look very optimally fitted with the simple linear regression, except for the tv & sales model. However, even the tv & sales model do not have a very good fitted linear regression model. The R2 explains about 64.5% of the data points in the training set with MSE about 10.5, but only explains about 50.7% of the datapoints in the testing set with a similar MSE about 10.7. This could tells us there is a linear relationship between the two vairables, in fact it is a strong positive relationship(sqrt of 0.644533 is 0.8028281261 which is the R).

Very interesting to see that on the radio & sales model, the R2 is higher on testing set(around 0.433) than on the training set(about 0.300). I search online and found 1 explaination that this could be due to the train-test-split percetage. Or it could be there are too many noises in the training set than in the testing set. I made a copy of this dataset, and splited the set into 50-50 for train and test, the new difference reduced from 13 to 4 in the R2 values. Along with the MSE differences reduced from 8.3 to 2.4).

As for the newspaper, this is likely to be the result of an underfitting model, to explain why both the train and test set have very low R2 scores(train: 0.058; test: 0.033), along with high MSE rates in both of the sets, meaning that the simple linear regression model is not the correct model to use for this set of data.

3.0.3 3. 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.

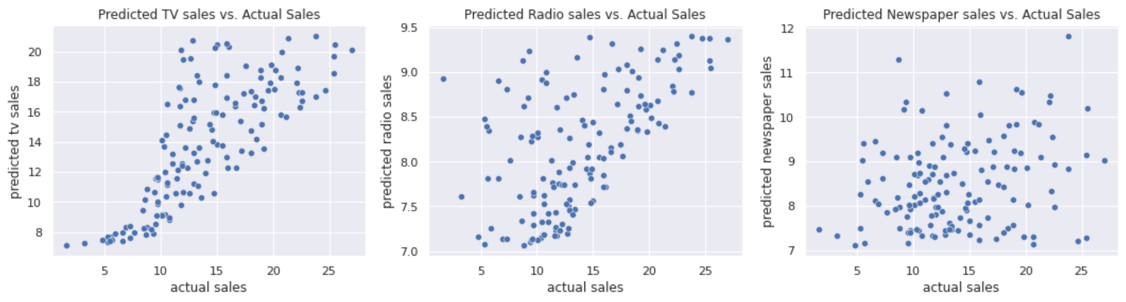
- 3.1 For each of the three regression models, plot predicted sales ($\hat{y}$) on the vertical axis, and actual sales (y) on the horizontal axis. Make sure both axes use the same scale. Comment on your observations. What would you expect this plot to look like for a model that explains the data well?
- 3.2 For each of the three regression models, compute the residuals ($y - \hat{y}$). Note that some of these will be negative, and some will be positive. What is the mean residual for each of the regression models? What *should* be the mean residual for a fitted linear regression model? Explain your answer.
- 3.3 For each of the three regression models, plot the residuals ($y - \hat{y}$) on the vertical axis, and actual sales (y) on the horizontal axis. Use the same scale for all three subplots. 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?
- 3.4 For each of the three regression models AND each of the three features, plot 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. 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, in addition to the text cells in which you write your comments, observations, and answers to the questions.

[29]:
```python
# 3.1
fig = plt.figure(figsize=(18,4))

plt.subplot(1,3,1)
sns.scatterplot(data=train, x=train['sales'], y=reg_tv.predict(train[['TV']]));
plt.title('Predicted TV sales vs. Actual Sales')
plt.xlabel('actual sales')
plt.ylabel('predicted tv sales')

plt.subplot(1,3,2)
sns.scatterplot(data=train, x=train['sales'], y=reg_tv.
 ↪predict(train[['radio']]));
plt.title('Predicted Radio sales vs. Actual Sales')
plt.xlabel('actual sales')
plt.ylabel('predicted radio sales')

plt.subplot(1,3,3)
```

```
sns.scatterplot(data=train, x=train['sales'], y=reg_tv.
 ↪predict(train[['newspaper']]));
plt.title('Predicted Newspaper sales vs. Actual Sales')
plt.xlabel('actual sales')
plt.ylabel('predicted newspaper sales')

plt.show()

#comments: What would you expect this plot to look like for a model that␣
 ↪explains the data well?
# If the model predicts really well, we should be expecting a very clear␣
 ↪diagonal line from (0,0) to (max(x_axis), max(y_axis))
```

```
[29]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5be201150>

[29]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5be201150>

[29]: Text(0.5, 1.0, 'Predicted TV sales vs. Actual Sales')

[29]: Text(0.5, 0, 'actual sales')

[29]: Text(0, 0.5, 'predicted tv sales')

[29]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5be25e610>

[29]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5be25e610>

[29]: Text(0.5, 1.0, 'Predicted Radio sales vs. Actual Sales')

[29]: Text(0.5, 0, 'actual sales')

[29]: Text(0, 0.5, 'predicted radio sales')

[29]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5b5e49b10>

[29]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5b5e49b10>

[29]: Text(0.5, 1.0, 'Predicted Newspaper sales vs. Actual Sales')

[29]: Text(0.5, 0, 'actual sales')

[29]: Text(0, 0.5, 'predicted newspaper sales')
```

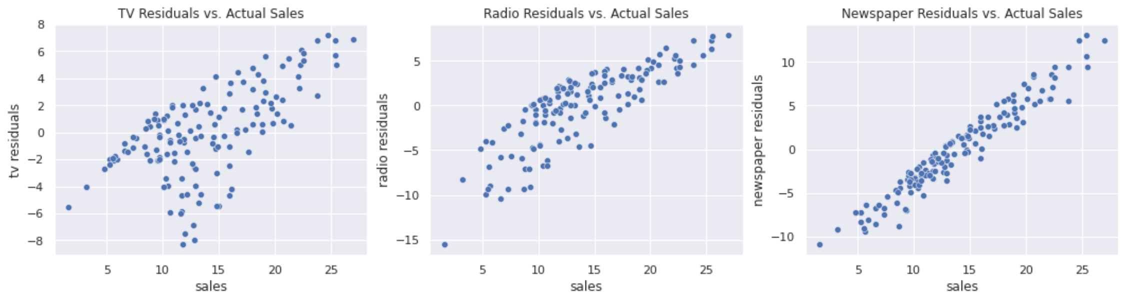


[32]:
```
# 3.2
import statistics as sts
df3 = pd.DataFrame([])

y_train_hat_tv = reg_tv.predict(train[['TV']])
y_train_hat_radio = reg_radio.predict(train[['radio']])
y_train_hat_newspaper = reg_news.predict(train[['newspaper']])

df3['residual_train_tv'] = train['sales'] - y_train_hat_tv
df3['residual_train_radio'] = train['sales'] - y_train_hat_radio
df3['residual_train_newspaper'] = train['sales'] - y_train_hat_newspaper

print("mean of tv residual: ", sts.mean(df3['residual_train_tv']))
print("mean of radio residual: ", sts.mean(df3['residual_train_radio']))
print("mean of newspaper residual: ", sts.mean(df3['residual_train_newspaper']))

#comments: What is the mean residual for each of the regression models? # see␣
 ↪print
# What should be the mean residual for a fitted linear regression model?
# the best scenario is that the mean of residual equals 0 or very close to 0,␣
 ↪that means there are no residuals, and the model fits very well
```

```
mean of tv residual:  -1.45915026093592e-15
mean of radio residual:  -3.3623897317219026e-16
mean of newspaper residual:  -8.310812355765458e-16
```

[35]:
```
# 3.3
fig = plt.figure(figsize=(18,4))

plt.subplot(1,3,1)
sns.scatterplot(data=train, x=train['sales'], y=df3['residual_train_tv']);
plt.title('TV Residuals vs. Actual Sales')
plt.xlabel('sales')
plt.ylabel('tv residuals')

plt.subplot(1,3,2)
sns.scatterplot(data=train, x=train['sales'], y=df3['residual_train_radio']);
plt.title('Radio Residuals vs. Actual Sales')
plt.xlabel('sales')
plt.ylabel('radio residuals')

plt.subplot(1,3,3)
sns.scatterplot(data=train, x=train['sales'],␣
 ↪y=df3['residual_train_newspaper']);
plt.title('Newspaper Residuals vs. Actual Sales')
plt.xlabel('sales')
plt.ylabel('newspaper residuals')
plt.show()

#comments: 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?
# the tv residuals show a weird upside-down v-shape pattern that do not look␣
↪like linear
# the other twos look like linear, especially the newspaper ones
```

```
[35]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5b2be75d0>
[35]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5b2be75d0>
[35]: Text(0.5, 1.0, 'TV Residuals vs. Actual Sales')
[35]: Text(0.5, 0, 'sales')
[35]: Text(0, 0.5, 'tv residuals')
[35]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5bdfb4f90>
[35]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5bdfb4f90>
[35]: Text(0.5, 1.0, 'Radio Residuals vs. Actual Sales')
[35]: Text(0.5, 0, 'sales')
[35]: Text(0, 0.5, 'radio residuals')
[35]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5be1f7ad0>
[35]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5be1f7ad0>
[35]: Text(0.5, 1.0, 'Newspaper Residuals vs. Actual Sales')
[35]: Text(0.5, 0, 'sales')
[35]: Text(0, 0.5, 'newspaper residuals')
```

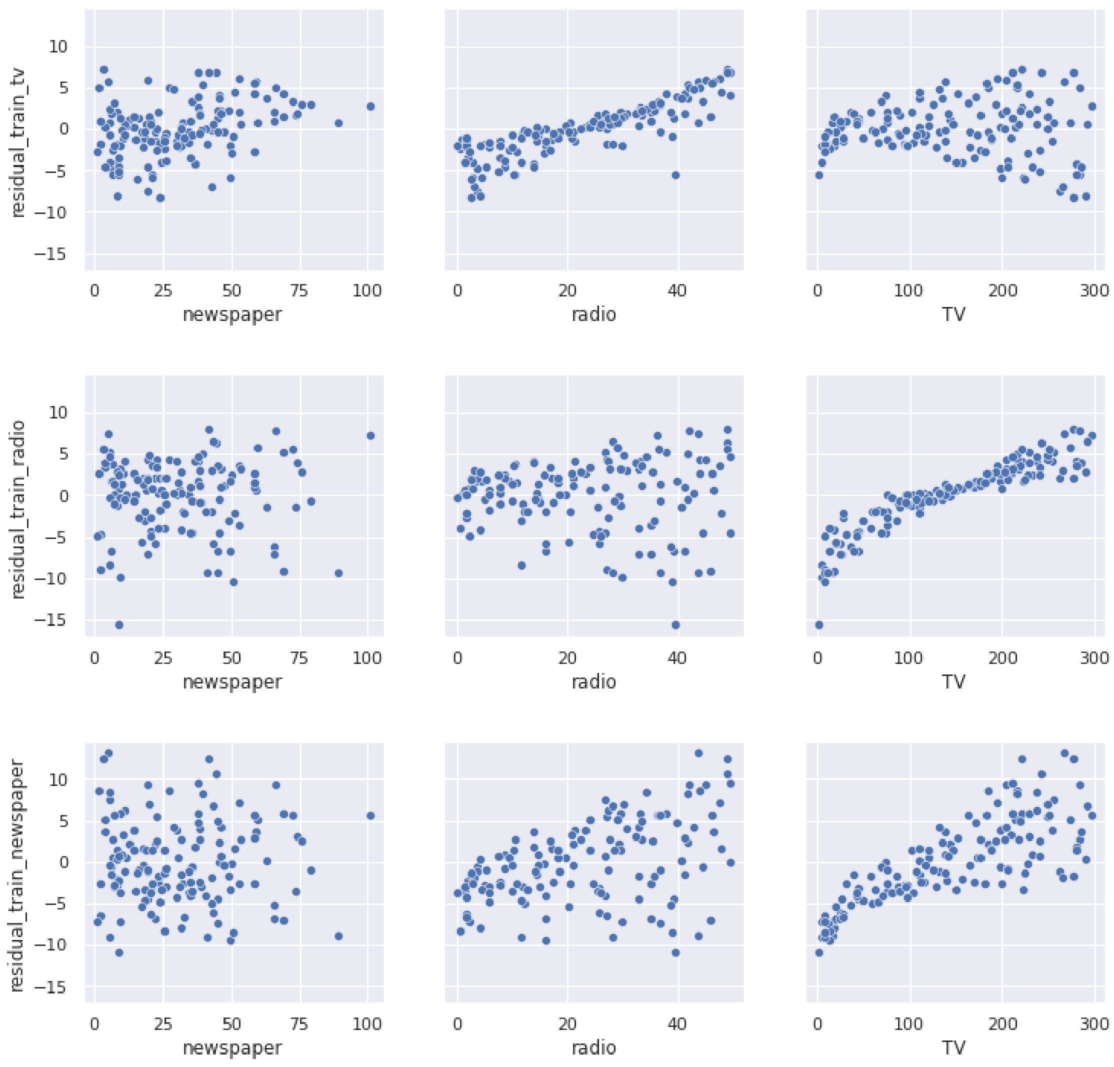


```
[36]: # 3.4

fig, axs = plt.subplots(nrows=3, ncols=3, figsize=(12,12), sharey=True)
plt.subplots_adjust(hspace=.4)
```

```
sns.scatterplot(data=train, x=train['TV'], y=df3['residual_train_tv'],␣
 ↪ax=axs[0,2]);

sns.scatterplot(data=train, x=train['TV'], y=df3['residual_train_radio'],␣
 ↪ax=axs[1,2]);

sns.scatterplot(data=train, x=train['TV'], y=df3['residual_train_newspaper'],␣
 ↪ax=axs[2,2]);

sns.scatterplot(data=train, x=train['radio'], y=df3['residual_train_tv'],␣
 ↪ax=axs[0,1]);

sns.scatterplot(data=train, x=train['radio'], y=df3['residual_train_radio'],␣
 ↪ax=axs[1,1]);

sns.scatterplot(data=train, x=train['radio'],␣
 ↪y=df3['residual_train_newspaper'], ax=axs[2,1]);

sns.scatterplot(data=train, x=train['newspaper'], y=df3['residual_train_tv'],␣
 ↪ax=axs[0,0]);

sns.scatterplot(data=train, x=train['newspaper'],␣
 ↪y=df3['residual_train_radio'], ax=axs[1,0]);

sns.scatterplot(data=train, x=train['newspaper'],␣
 ↪y=df3['residual_train_newspaper'], ax=axs[2,0]);

plt.show()


#comment : what would you expect this plot to look like for a model that␣
 ↪explains the data well?
# I think this should be telling that, the regression residuals on tv compare␣
 ↪to the units in newspaper, show a pattern that in the lower units of␣
 ↪newspaper
# residuals appear to be more clustered than in the higher units of newspaper,␣
 ↪this means that, based on this linear model,
# spendings in the lower units of tv do not perform as good as in newspaper,␣
 ↪therefore lower spending should be directed on newspaper instead of tv
# lower right corner shows x-axis on tv, clearly showing a pattern as tv in␣
 ↪higher units perform better
# similar interpretations could apply to the other charts, as the cahrt tells␣
 ↪the performance of 1 variable in lower or higher or middle units whether it␣
 ↪outperforms
# or underperforms the ther variable, therefore, the spedning could be directed␣
 ↪more wisely
```

```
# as for the self-self comparison, it's showing the residuals pattern from the␣
↪regression model, to indicate whether the model
# is fitted or not at all
# for instance, the residuals in both tv and radio, appear to have a more␣
↪fit-to-the-mean type of pattern rather than diagonal linear pattern
```

```
[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5b2a6af50>

[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5bdf8d850>

[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5bdfc91d0>

[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5b49997d0>

[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5bdfaf690>

[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5b5edf090>

[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5b2bb4f50>

[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5b2be7290>

[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5b4b137d0>
```



3.0.4 4. 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 should add text cells in which you write your 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:

- 4.1 Plot predicted sales ($\hat{y}$) on the vertical axis, and actual sales (y) on the horizontal axis. Make sure both axes use the same scale. Comment on your observations. What would you

expect this plot to look like for a model that explains the data well?
- 4.2 Compute the residuals $(y - \hat{y})$. What is the mean of the residuals? What *should* be the mean of the residuals for a fitted linear regression model? Explain your answer.
- 4.3 Plot 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?
- 4.4 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?

The code in the last part of this section is not provided for you. You will need to write code, in addition to the text cells in which you write your comments, observations, and answers to the questions.

Fit a multiple linear regression

[37]:
```
reg_multi = LinearRegression().fit(train[['TV', 'radio', 'newspaper']],␣
 ↪train['sales'])
```

Look at coefficients

[38]:
```
print("Coefficients (TV, radio, newspaper):", reg_multi.coef_)
print("Intercept: ", reg_multi.intercept_)
```

```
Coefficients (TV, radio, newspaper): [0.04580858 0.18955803 0.00223433]
Intercept:  2.781641375973688
```

Compute R2, MSE for multiple regression

[39]:
```
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:   0.8904262962439081
Multiple regression MSE:  2.9752401868969773
```

[40]:
```
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:   0.911794520968297
Multiple regression MSE:  2.360943133954173
```

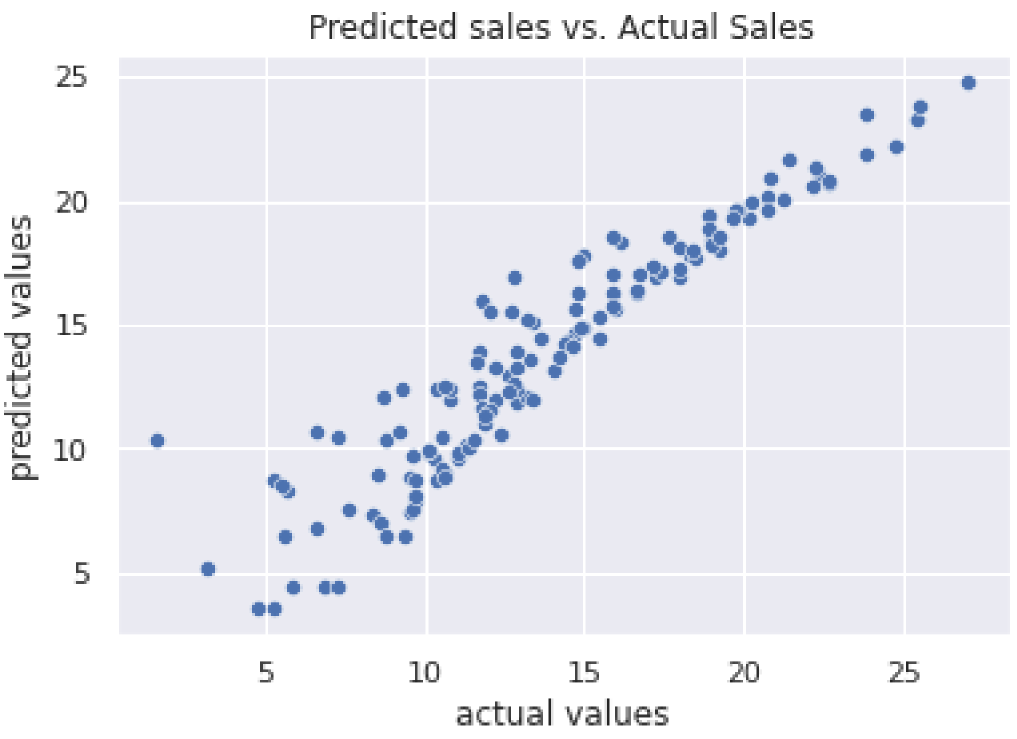
[41]:
```
# 4.1

sns.scatterplot(data=train, x=train['sales'], y=y_pred_tr_multi)
plt.title('Predicted sales vs. Actual Sales')
plt.xlabel('actual values')
plt.ylabel('predicted values')
plt.show()

#comments: What would you expect this plot to look like for a model that␣
↪explains the data well?
# If the model predicts really well, we should be expecting a very clear␣
↪diagonal line from (0,0) to (max(x_axis), max(y_axis))
# this model shows a linear relationship between the two variables
```

[41]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb5b2ac6510>
```

[41]:
```
Text(0.5, 1.0, 'Predicted sales vs. Actual Sales')
```

[41]:
```
Text(0.5, 0, 'actual values')
```

[41]:
```
Text(0, 0.5, 'predicted values')
```



[42]:
```
# 4.2

multi_residual = train['sales'] - y_pred_tr_multi
print("mean of residual: ", sts.mean(multi_residual))

#comments:  What is the mean of the residuals? #see print
#What should be the mean of the residuals for a fitted linear regression model?
# the best scenario is that the mean of residual equals 0 or very close to 0,␣
 ↪that means there are no residuals, and the model fits very well
# this residual shows the model is well fitted
```

```
mean of residual:  -3.29260428445975e-15
```

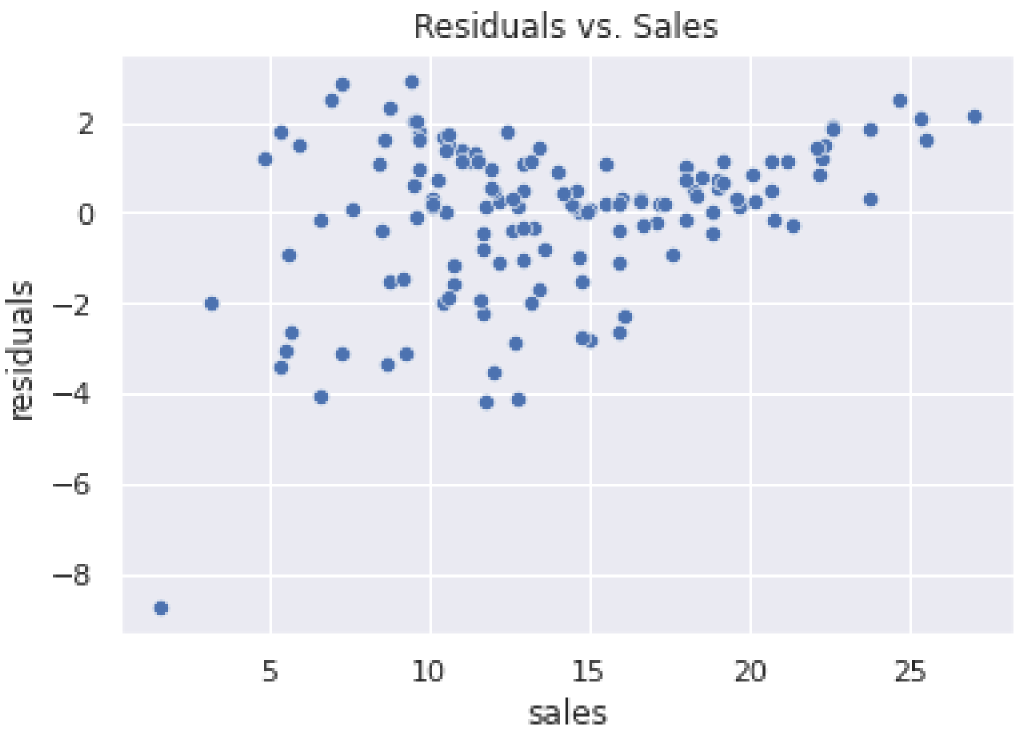
[43]:
```
# 4.3

sns.scatterplot(data=train, x=train['sales'], y=multi_residual)
plt.title('Residuals vs. Sales')
plt.xlabel('sales')
plt.ylabel('residuals')
plt.show()

#comments: 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?
# it seems that the model predicts well in ther lower values compare to the␣
 ↪higher values in sales
# among 10-20 units of sales, cluster the largest amount of residuals
```

[43]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb5b2c1de10>
```

[43]:
```
Text(0.5, 1.0, 'Residuals vs. Sales')
```

[43]:
```
Text(0.5, 0, 'sales')
```

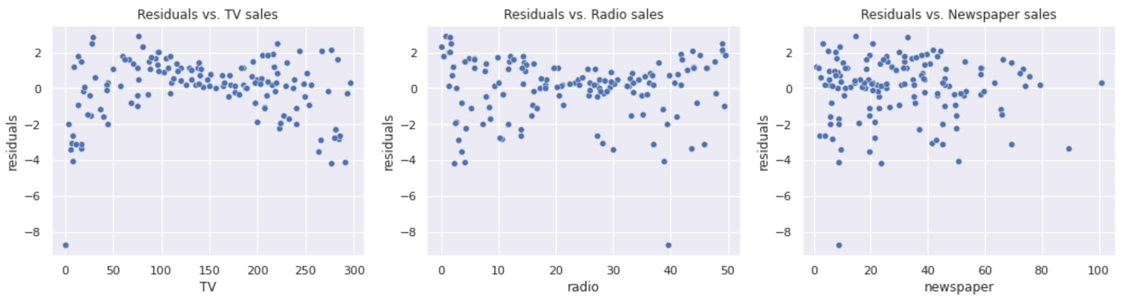
[43]:
```
Text(0, 0.5, 'residuals')
```



[44]:
```
# 4.4

fig = plt.figure(figsize=(18,4))

plt.subplot(1,3,1)
sns.scatterplot(data=train, x=train['TV'], y=multi_residual);
plt.title('Residuals vs. TV sales')
plt.xlabel('TV')
plt.ylabel('residuals')

plt.subplot(1,3,2)
sns.scatterplot(data=train, x=train['radio'], y=multi_residual);
plt.title('Residuals vs. Radio sales')
plt.xlabel('radio')
plt.ylabel('residuals')

plt.subplot(1,3,3)
sns.scatterplot(data=train, x=train['newspaper'], y=multi_residual);
plt.title('Residuals vs. Newspaper sales')
plt.xlabel('newspaper')
plt.ylabel('residuals')

plt.show()
```

```
#comments: 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?
# residuals on newspaper appear to cluster in the lower values units of␣
 ↪newspaper, and more scattered in the higher values units
```

```
[44]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5be152c90>
[44]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5be152c90>
[44]: Text(0.5, 1.0, 'Residuals vs. TV sales')
[44]: Text(0.5, 0, 'TV')
[44]: Text(0, 0.5, 'residuals')
[44]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5be135390>
[44]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5be135390>
[44]: Text(0.5, 1.0, 'Residuals vs. Radio sales')
[44]: Text(0.5, 0, 'radio')
[44]: Text(0, 0.5, 'residuals')
[44]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5be020a10>
[44]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5be020a10>
[44]: Text(0.5, 1.0, 'Residuals vs. Newspaper sales')
[44]: Text(0.5, 0, 'newspaper')
[44]: Text(0, 0.5, 'residuals')
```



3.0.5 5. 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 (4) suggest that a model including an interaction effect may explain sales data better than a model including additive effects. Add four columns to your data frame:

- `newspaper` × `radio`
- `TV` × `radio`
- `newspaper` × `TV`
- `newspaper` × `radio` × `TV`

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 (4) for the model including interaction effects. Comment on the results. 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?

(If you think the results suggest further opportunities for improving the model, you are welcome to try and to comment on the results!)

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

[45]:
```
# create interaction effects

train2 = train.copy()
# train2['newspaper_radio'] = pd.Series(train['newspaper'] * train['radio'],␣
↪name='newspaper_radio')
# x_t = PolynomialFeatures(2, interaction_only=True, include_bias=False).
↪fit_transform(x)
train2['newspaper_radio'] = train2['newspaper'] * train2['radio']
train2['TV_radio'] = train2['TV'] * train2['radio']
train2['newspaper_TV'] = train2['newspaper'] * train2['TV']
train2['newspaper_radio_TV'] = train2['newspaper'] * train2['radio'] *␣
↪train2['TV']
train2
```

[45]:
```
        TV  radio  newspaper  ...  TV_radio  newspaper_TV  newspaper_radio_TV
73    26.8   33.0       19.3  ...    884.40        517.24           17068.920
102  296.4   36.3      100.9  ...  10759.32      29906.76         1085615.388
98   184.9   21.0       22.0  ...   3882.90       4067.80           85423.800
2     44.5   39.3       45.1  ...   1748.85       2006.95           78873.135
143  220.5   33.2       37.9  ...   7320.60       8356.95          277450.740
..     ...    ...        ...  ...       ...           ...                 ...
92    28.6    1.5       33.0  ...     42.90        943.80            1415.700
164  163.5   36.8        7.4  ...   6016.80       1209.90           44524.320
147  240.1    7.3        8.7  ...   1752.73       2088.87           15248.751
90   109.8   47.8       51.4  ...   5248.44       5643.72          269769.816
78   120.5   28.5       14.2  ...   3434.25       1711.10           48766.350

[140 rows x 8 columns]
```

[54]:
```
# multiple linear regression model for 7 features
```

```
reg_multi2 = LinearRegression().fit(train2[['TV', 'radio', 'newspaper',␣
 ↪'newspaper_radio',        'TV_radio',        'newspaper_TV',        'newspaper_radio_TV']],
 ↪train2['sales'])
```

[47]:
```
# Coefficient and Interception

print("Coefficients (TV, radio, newspaper,␣
 ↪newspaper_radio,        TV_radio,        newspaper_TV, newspaper_radio_TV):
 ↪\n", reg_multi2.coef_)
print("Intercept: ", reg_multi2.intercept_)
```

```
Coefficients (TV, radio, newspaper, newspaper_radio,    TV_radio,
newspaper_TV, newspaper_radio_TV):
 [ 2.02909529e-02  7.99102575e-04  1.85063230e-02  2.75653765e-04
  1.26256298e-03 -8.88358073e-05 -1.75649072e-06]
Intercept:  6.432347971552468
```

[48]:
```
# R2 and MSE for multiple linear regression

y_pred_tr_multi2 = reg_multi2.predict(train2[['TV', 'radio', 'newspaper',␣
 ↪'newspaper_radio',        'TV_radio',        'newspaper_TV',        'newspaper_radio_TV']])

r2_tr_multi2  = metrics.r2_score(train2['sales'], y_pred_tr_multi2)
mse_tr_multi2 = metrics.mean_squared_error(train2['sales'], y_pred_tr_multi2)

print("Multiple regression R2:  ", r2_tr_multi2)
print("Multiple regression MSE: ", mse_tr_multi2)


# Multiple regression R2:   0.9018262100622871
# Multiple regression MSE:  2.885891634586171
```

```
Multiple regression R2:   0.9646498549423631
Multiple regression MSE:  0.9598577814092687
```

[49]:
```
# Replicate 4.1

sns.scatterplot(data=train2, x=train['sales'], y=y_pred_tr_multi2)
plt.title('Predicted sales vs. Actual Sales')
plt.xlabel('actual values')
plt.ylabel('predicted values')
plt.show()

#comments: What would you expect this plot to look like for a model that␣
 ↪explains the data well?
# If the model predicts really well, we should be expecting a very clear␣
 ↪diagonal line from (0,0) to (max(x_axis), max(y_axis))
```

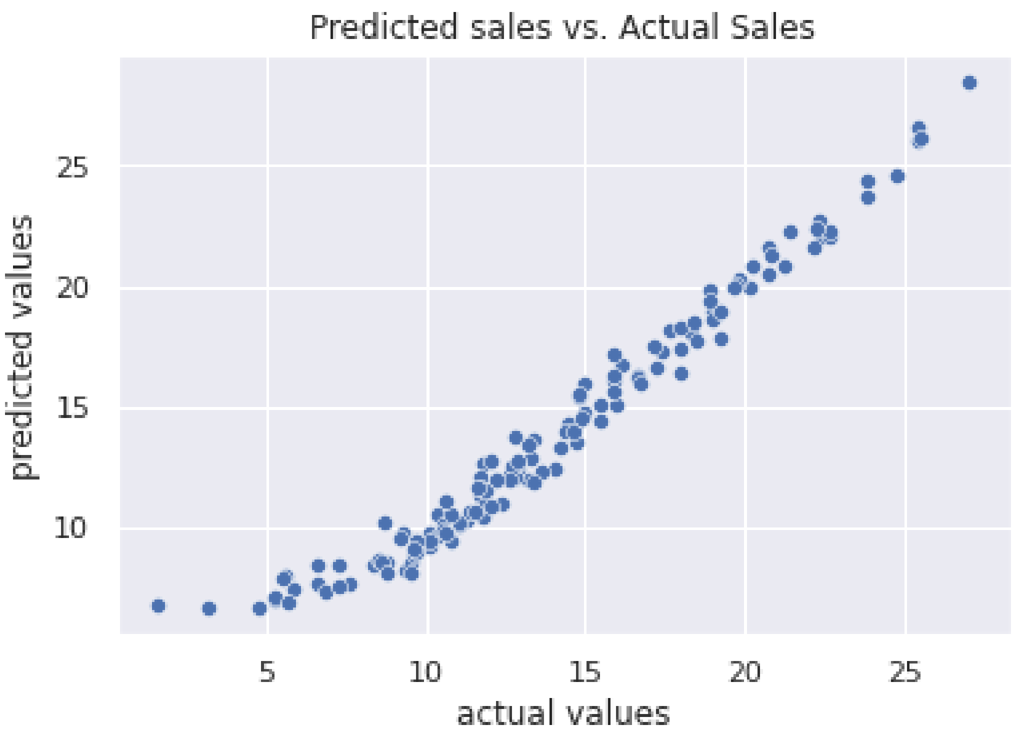
```
# this model shows a linear relationship between the two variables, and almost␣
↪a very clear diagonal line, but with a little tilted from the lower values␣
↪units
```

[49]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb5b2b23790>
```

[49]:
```
Text(0.5, 1.0, 'Predicted sales vs. Actual Sales')
```

[49]:
```
Text(0.5, 0, 'actual values')
```

[49]:
```
Text(0, 0.5, 'predicted values')
```



[50]:
```
# Replicate 4.2

multi_residual2 = train2['sales'] - y_pred_tr_multi2
print("mean of residual: ", sts.mean(multi_residual2))

#comments:  What is the mean of the residuals? #see print
#What should be the mean of the residuals for a fitted linear regression model?
# the best scenario is that the mean of residual equals 0 or very close to 0,␣
↪that means there are no residuals, and the model fits very well
# this residual shows the model is well fitted
```

```
mean of residual:  -2.011089707463855e-15
```

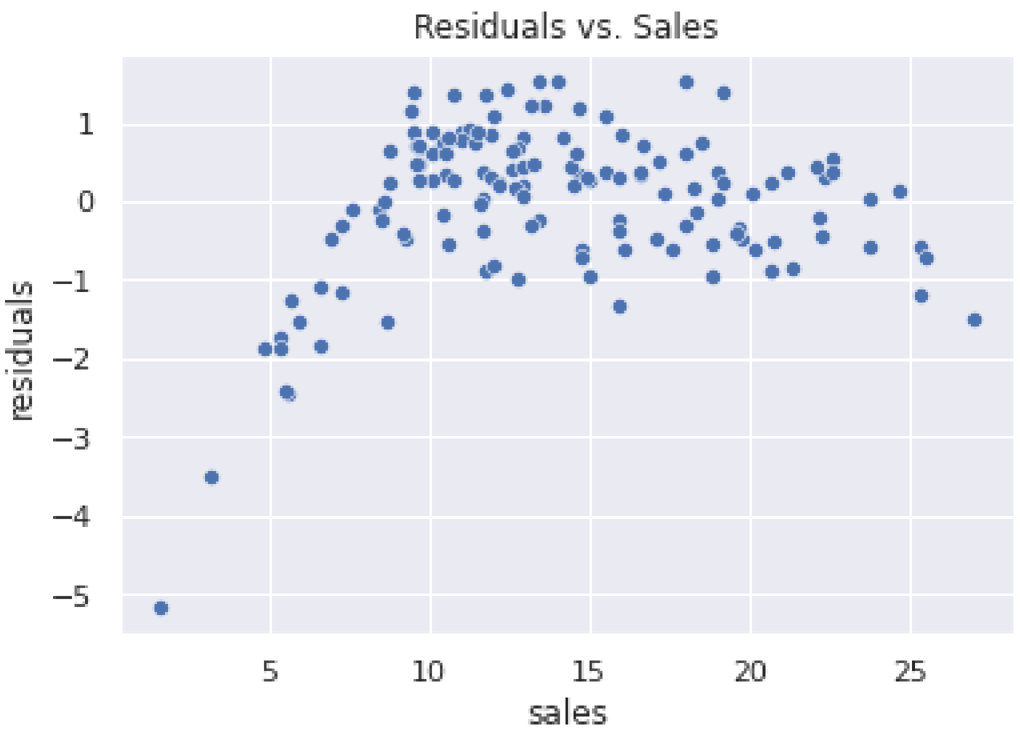
[51]:
```
# Replicate 4.3

sns.scatterplot(data=train2, x=train2['sales'], y=multi_residual2)
plt.title('Residuals vs. Sales')
plt.xlabel('sales')
plt.ylabel('residuals')
plt.show()

#comments: 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?
# it seems that the model predicts well in ther lower values compare to the␣
↪higher values in sales
# among 10-20 units of sales, cluster the largest amount of residuals
```

[51]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb5be152ad0>
```

[51]:
```
Text(0.5, 1.0, 'Residuals vs. Sales')
```

[51]:
```
Text(0.5, 0, 'sales')
```

[51]:
```
Text(0, 0.5, 'residuals')
```



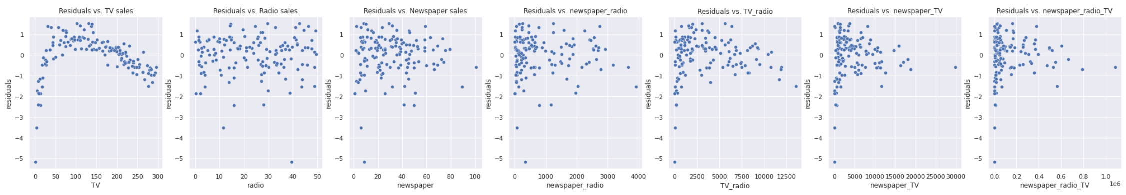
[62]:
```
# Replicate 4.4

fig = plt.figure(figsize=(35,5))
```

```
plt.subplot(1,7,1)
sns.scatterplot(data=train2, x=train2['TV'], y=multi_residual2);
plt.title('Residuals vs. TV sales')
plt.xlabel('TV')
plt.ylabel('residuals')

plt.subplot(1,7,2)
sns.scatterplot(data=train2, x=train2['radio'], y=multi_residual2);
plt.title('Residuals vs. Radio sales')
plt.xlabel('radio')
plt.ylabel('residuals')

plt.subplot(1,7,3)
sns.scatterplot(data=train2, x=train2['newspaper'], y=multi_residual2);
plt.title('Residuals vs. Newspaper sales')
plt.xlabel('newspaper')
plt.ylabel('residuals')

plt.subplot(1,7,4)
sns.scatterplot(data=train2, x=train2['newspaper_radio'], y=multi_residual2);
plt.title('Residuals vs. newspaper_radio')
plt.xlabel('newspaper_radio')
plt.ylabel('residuals')

plt.subplot(1,7,5)
sns.scatterplot(data=train2, x=train2['TV_radio'], y=multi_residual2);
plt.title('Residuals vs. TV_radio')
plt.xlabel('TV_radio')
plt.ylabel('residuals')

plt.subplot(1,7,6)
sns.scatterplot(data=train2, x=train2['newspaper_TV'], y=multi_residual2);
plt.title('Residuals vs. newspaper_TV')
plt.xlabel('newspaper_TV')
plt.ylabel('residuals')

plt.subplot(1,7,7)
sns.scatterplot(data=train2, x=train2['newspaper_radio_TV'], y=multi_residual2);
plt.title('Residuals vs. newspaper_radio_TV')
plt.xlabel('newspaper_radio_TV')
plt.ylabel('residuals')

plt.show()

#comments: 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?
```

```
[62]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5aab2f7d0>
[62]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5aab2f7d0>
[62]: Text(0.5, 1.0, 'Residuals vs. TV sales')
[62]: Text(0.5, 0, 'TV')
[62]: Text(0, 0.5, 'residuals')
[62]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5aab46990>
[62]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5aab46990>
[62]: Text(0.5, 1.0, 'Residuals vs. Radio sales')
[62]: Text(0.5, 0, 'radio')
[62]: Text(0, 0.5, 'residuals')
[62]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5aaabfd10>
[62]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5aaabfd10>
[62]: Text(0.5, 1.0, 'Residuals vs. Newspaper sales')
[62]: Text(0.5, 0, 'newspaper')
[62]: Text(0, 0.5, 'residuals')
[62]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5aaadf5d0>
[62]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5aaadf5d0>
[62]: Text(0.5, 1.0, 'Residuals vs. newspaper_radio')
[62]: Text(0.5, 0, 'newspaper_radio')
[62]: Text(0, 0.5, 'residuals')
[62]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5aaa1d950>
[62]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5aaa1d950>
[62]: Text(0.5, 1.0, 'Residuals vs. TV_radio')
[62]: Text(0.5, 0, 'TV_radio')
[62]: Text(0, 0.5, 'residuals')
[62]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5aa9b1e10>
[62]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5aa9b1e10>
[62]: Text(0.5, 1.0, 'Residuals vs. newspaper_TV')
[62]: Text(0.5, 0, 'newspaper_TV')
[62]: Text(0, 0.5, 'residuals')
[62]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5aa95d6d0>
[62]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5aa95d6d0>
[62]: Text(0.5, 1.0, 'Residuals vs. newspaper_radio_TV')
```

[62]:
```
Text(0.5, 0, 'newspaper_radio_TV')
```

[62]:
```
Text(0, 0.5, 'residuals')
```


Residuals vs. TV sales
TV
residuals
Residuals vs. Radio sales
radio
residuals
Residuals vs. Newspaper sales
newspaper
residuals
Residuals vs. newspaper_radio
newspaper_radio
residuals
Residuals vs. TV_radio
TV_radio
residuals
Residuals vs. newspaper_TV
newspaper_TV
residuals
Residuals vs. newspaper_radio_TV
newspaper_radio_TV
residuals
1e6