cccc

1.a *As it is given in that we have two vectors $(y_1, y_2...., y_n)$ and $(x_1, x_2...., x_n)$ of length n. Now considering that X is a matrix of length n*(d+1) we cannot make the assumption that the given data set has a non zero mean which means that w_0 cannot be assumed to be zero in this case.So the X matrix shapes up*

$$\begin{bmatrix} 1 & x_{11} & x_{12} & .... & x_{1d} \\ 1 & x_{21} & x_{22} & .... & x_{2d} \\ 1 & x_{31} & x_{32} & .... & x_{3d} \\ . & .... & .... & .... & .... \\ 1 & x_{n1} & x_{n2} & ....1 & x_{nd} \end{bmatrix}$$

*As it is clear that the matrix W(weight) should be dimension (d+1)*n. We can say that the weight matrix may look like:*

$$\begin{bmatrix} w_0 & w_0 & w_0 & .... & w_0 \\ w_{11} & x_{12} & x_{13} & .... & x_{1n} \\ w_{21} & x_{22} & x_{33} & .... & x_{2n} \\ . & .... & .... & .... & .... \\ w_{d1} & x_{d2} & w_{d3} & .... & x_{dn} \end{bmatrix}$$

Since the model that we are using to fit this data set is a multivariate linear regression model,so as to maintain the property of matrix multiplication of matrix X and W, so the matrix w had to be of the shape mentioned above.Now as the model used is a linear regression we can say that the predicted value of the model can be of the type:

$$\widetilde{y} = Xw$$

It is evident that y is a square matrix of dimension (d+1)(d+1).As we know that the mean square error is the square of the difference between the expected and the predicted value over all the data points.This can be written as:*

$$MSE(w) = \frac{\sum_{i=1}^{n}(y - Xw)^2}{n}$$

Now this looks similar to the the L2 norm of the term y-Xw ,if we take the square of the L2 norm of $y - \widetilde{y}$ we get:

$$\sum_{i=1}^{n} ||y - Xw||\,|_2^2$$

$$= \sum_{i=1}^{n} \sqrt{(y - Xw)^2})^2 = \sum_{i=1}^{n}(y - Xw)^2$$

This result is equivalent to MSE of the function $y - \widetilde{y}$.

1.b *For getting the optimal values of MSE(w), we can calculate the gradient or the partial derivative w.r.t w . As derived from the above problem we have the MSE(w) as*

$$MSE(w) = \frac{1}{n}(y - Xw)^T(y - Xw)$$
$$= \frac{1}{n}(y^T - X^T w^T)((y - Xw))$$
$$= \frac{1}{n}(y^T y - X^T w^T y - y^T X w - X^T X w^T w)$$

Taking gradient of MSE we get:

$$\nabla(MSE) = \frac{\partial MSE}{\partial w}$$
$$= 2X^T X w - 2X^T y$$

For the optimum solution $\nabla(MSE) = 0$. Since it is assumed that $rank(x) = k(fullrank)$, then $X^T X$ is a positive definite and unique solution of the normal equation is

$$X^T X \widehat{w} = X^T y$$

$$\widehat{w} = X^T y (X^T X)^- 1$$

Few other assumptions that we need to consider is as follows :
(i) X is a non-stochastic matrix
(ii) X has to be a singular matrix to get its inverse
(iii)$\lim_{x \to +\infty}(\frac{X^T X}{n}) = \Delta$ *exists and is a non-stochastic and non singular matrix (with finite elements).*

Name: Farhan Rahman

NetID: fr2119

First Task is to import neseccary libraries.

In [ ]:

```
from sklearn import datasets
from sklearn import metrics
from sklearn import preprocessing
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()

# for 3d interactive plots
from ipywidgets import interact, fixed
from mpl_toolkits import mplot3d


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

For reading the data set "Advertising.csv", I used the link adress from the author's website as follows.

In [ ]:

```
url = 'http://faculty.marshall.usc.edu/gareth-james/ISL/Advertising.csv'
ds = pd.read_csv(url,index_col=0)
```

To get a brief overview of the type of data we are dealing with I used the following.The head() method display the first few data values.

In [ ]:

```
ds.head()
```

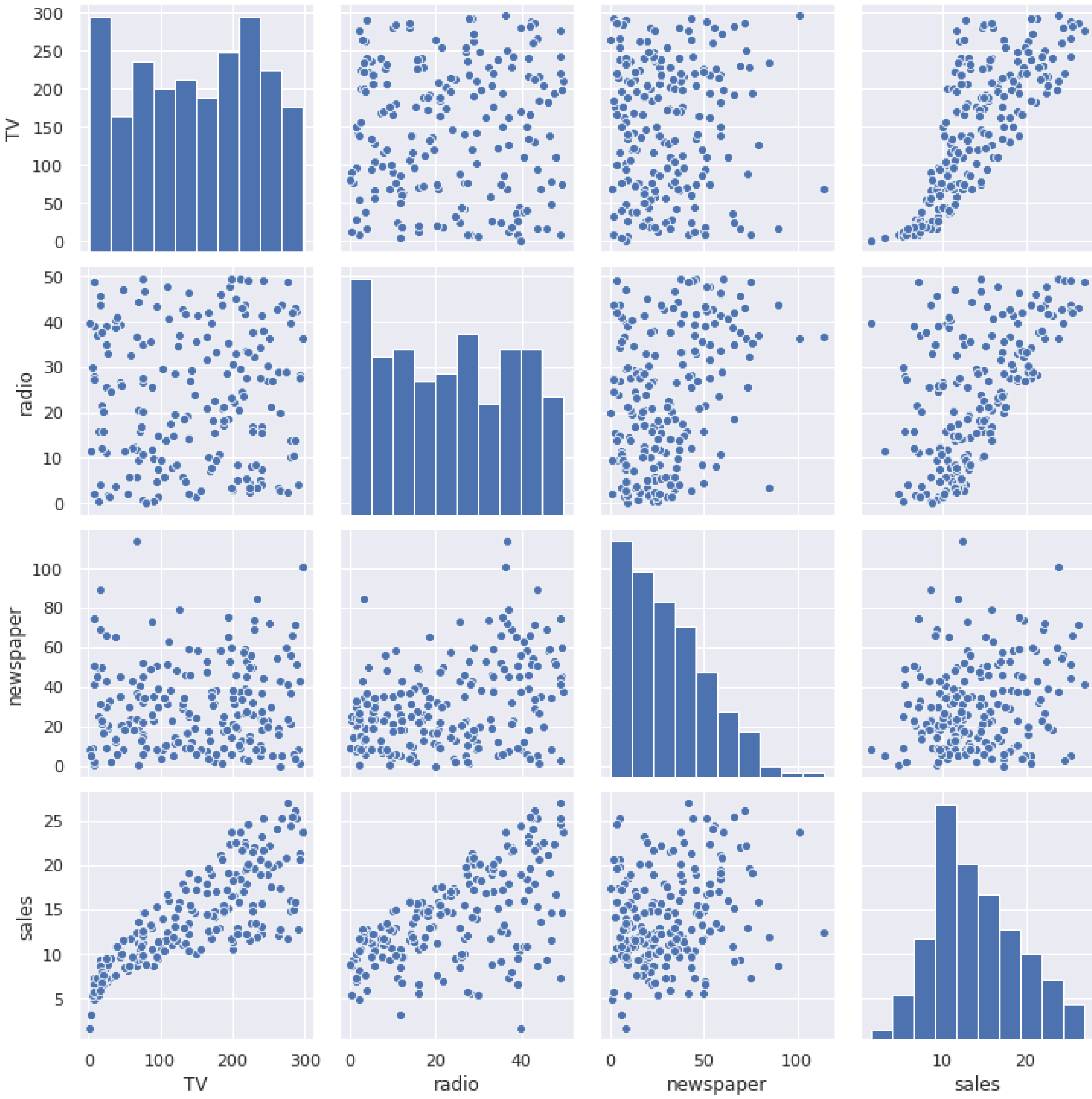
Out[ ]:

	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

Now to visualise the relation between the different features of our dataset we can use pairplot() which gives us the scater plot between two features and the histogram.

In [ ]:

```
sns.pairplot(ds);
```

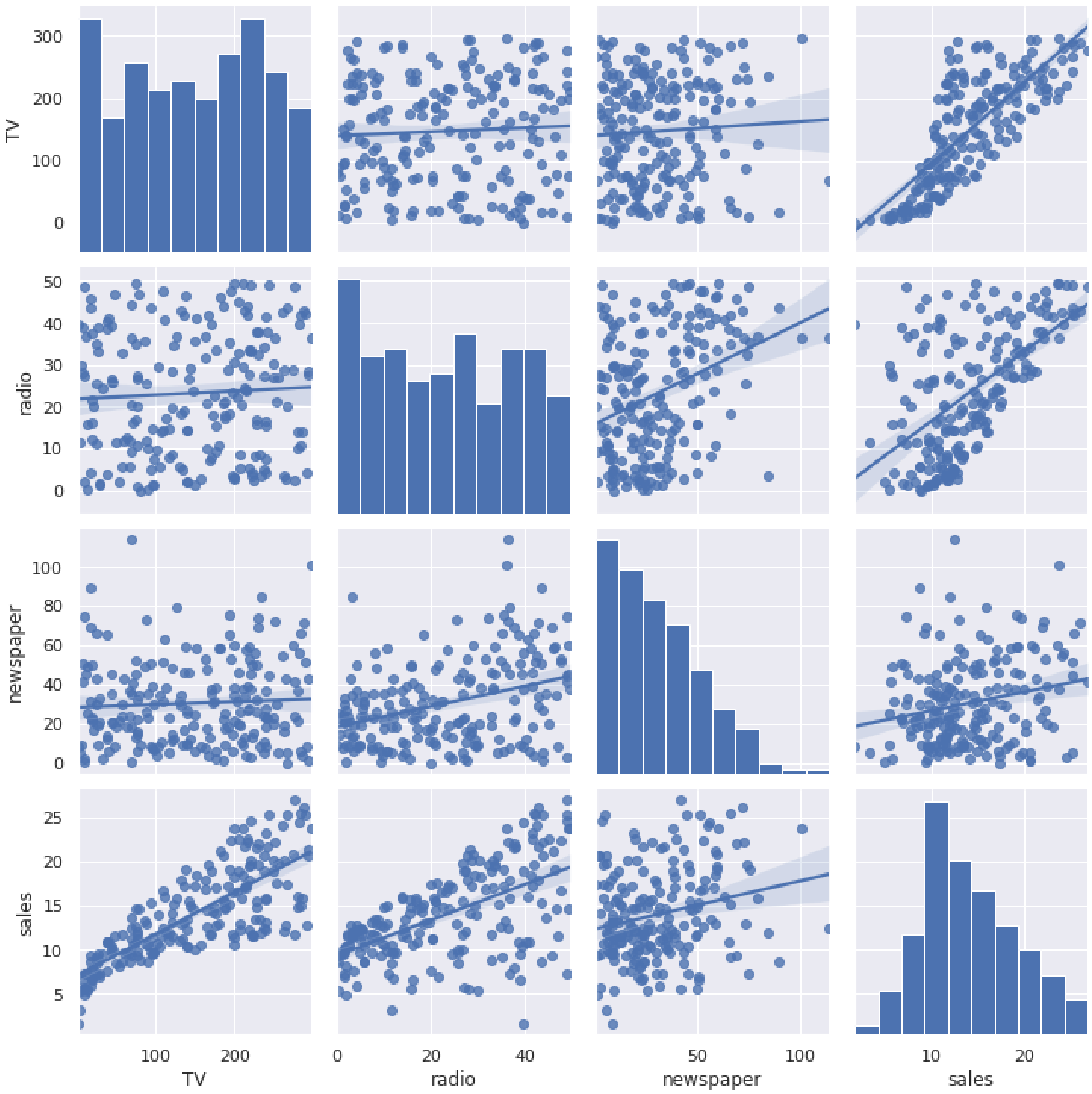


TV
radio
newspaper
sales

To see how a linear regression line fits all the relation (plots) we can use kind ="reg" using the seaborn library.

In [ ]:

```
g = sns.pairplot(ds, kind="reg")
```



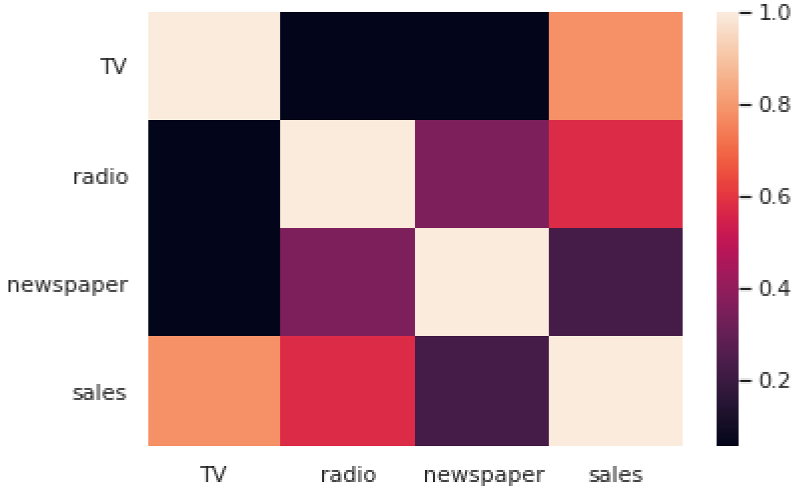
As it is evident that there many features that are correlated to each other we can use our common sense or analytical skills or ponder upon the .corr() function to that for us.

In [ ]:

```
corr = ds.corr()
sns.heatmap(corr)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7bcd4db438>
```



The heatmap gives us the extent to which these features are correlated with each other and espically focusing on the last row which is the relation between the sales and the advertising budget.It is quite clear that the sales of TV is positively correlated to the budget of advertising while the correlation between radio and sales is decreasing and it decreases even further for the the sales of newspaper and the budget of advertising.

Now to get into the ML part we have to fit regression models.We have to split the model into 70:30

In [ ]:

```
train, test = train_test_split(ds, test_size=0.3)
```

In [ ]:

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 140 entries, 150 to 56
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 [ ]:

```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 60 entries, 170 to 93
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
```

After splitting the data into required spilt we can proceed to train the model, althought the linear univarate model is not the best fit for all the features we can use it to see how it works with different features. For fitting the training data we can use the .fit function.

In [ ]:

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

For the intercept and coefficient values we can use the .intercept *and .coef*

In [ ]:

```
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:  0.04538800976455547 7.419155839124203
Radio:  0.20852085254440827 9.22127040209196
Newspaper:  0.04736353326313257 12.532951188982924
```

fitting the model for the test set.

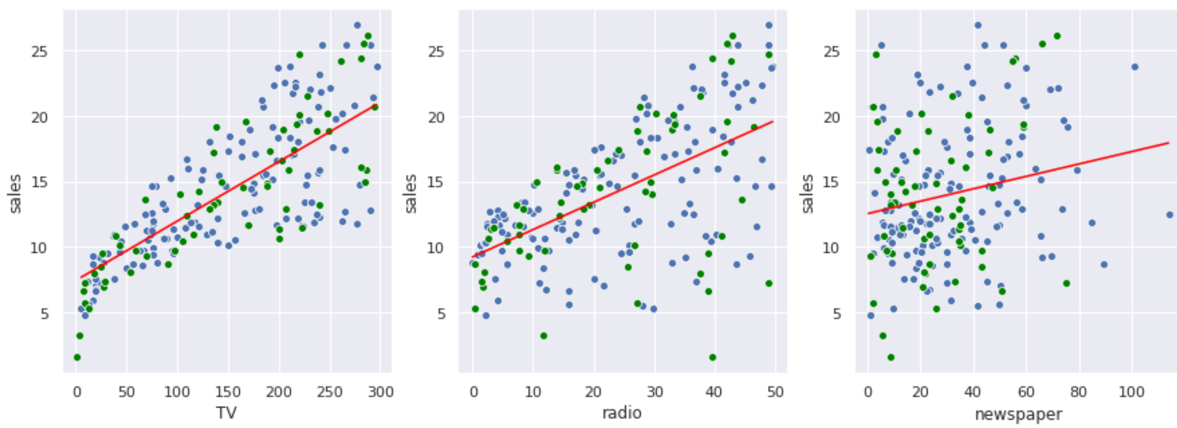
In [ ]:

```
y_pred_tv = reg_tv.predict(test[['TV']])
y_pred_radio = reg_radio.predict(test[['radio']])
y_pred_news = reg_news.predict(test[['newspaper']])
```

We can observe how the linear regressiong model fits our traning and test set using the following plots. The red blue points are the scatter plot for the training set while the green colored points are for the test data set .

In [ ]:

```
fig = plt.figure(figsize=(15,5))

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



Using the mathematical method to calculate the R2 score is as follows.

In [ ]:

```
r2_tv = 1-np.mean( (y_pred_tv - test['sales'])**2 / np.std(test['sales'])**2 )
r2_radio = 1-np.mean( (y_pred_radio - test['sales'])**2 / np.std(test['sales'])**2
)
r2_news = 1-np.mean( (y_pred_news - test['sales'])**2 / np.std(test['sales'])**2 )
print("TV: ", r2_tv)
print("Radio: ", r2_radio)
print("Newspaper: ", r2_news)
```

```
TV:  0.6898701888494395
Radio:  0.23131405832703855
Newspaper:  0.060792498280083374
```

Although the same can be achived by the inbult metrics parameter using metrics.mean_suarred_error and metrics.r2_score.

In [ ]:

```
print('Mean Squared Error for TV:', metrics.mean_squared_error(test['sales'], y_pr
ed_tv))
print('R2 score for TV:',(metrics.r2_score(test['sales'], y_pred_tv)))
print('Mean Squared Error for Radio:', metrics.mean_squared_error(test['sales'], y
_pred_radio))
print('R2 score for Radio:',(metrics.r2_score(test['sales'], y_pred_radio)))
print('Mean Squared Error for Newspaper:', metrics.mean_squared_error(test['sales'
], y_pred_news))
print('R2 score for Newspaper:',(metrics.r2_score(test['sales'], y_pred_news)))
```

```
Mean Squared Error for TV: 9.837820709167202
R2 score for TV: 0.6898701888494394
Mean Squared Error for Radio: 24.383965049282832
R2 score for Radio: 0.23131405832703844
Mean Squared Error for Newspaper: 29.793185557836335
R2 score for Newspaper: 0.06079249828008293
```

Apparently the best fit of a model is not classified depending on the R2 score or the mean squared error, but still it helps us to get a gist of what is going on with the model. As discussed in the classes that the the lesser the MSE the model fits the linear regression line better and the closest our R2 value to 1 , the better is our model. So keeping all this in mind we can say that the model for TV serves the best fit for the linear regression model, although just the R2 score and the MSE values are in no way adequate to say which models is the best.Now to understand this better we can use various plots to understand the exact relationship between the different features.

Various other plots: the plots below shows us the relation between the actual sales and the predicted sales. For the model that fits the best as a linear regression type we can expect that the relation between sales and ypredicted should be linear and highly correlated which is observed for the sales of TV while decreased correlation is observed between the other two which is radio and rewspaper.

In [ ]:

```
fig = plt.figure(figsize=(15,4))
plt.subplot(1,3,1)
sns.scatterplot(data=test, x="sales", y= y_pred_tv);
plt.xlim(0,30)
plt.ylim(0,30)

plt.subplot(1,3,2)
sns.scatterplot(data=test, x="sales", y= y_pred_radio);
plt.xlim(0,30)
plt.ylim(0,30)

plt.subplot(1,3,3)
sns.scatterplot(data=test, x="sales", y= y_pred_news);
plt.xlim(0,30)
plt.ylim(0,30)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7bd19aafd0>
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7bd19aafd0>
```

Out[ ]:

```
(0.0, 30.0)
```

Out[ ]:

```
(0.0, 30.0)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7bd19aae80>
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7bd19aae80>
```

Out[ ]:

```
(0.0, 30.0)
```

Out[ ]:

```
(0.0, 30.0)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7bd19aab38>
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7bd19aab38>
```

Out[ ]:

```
(0.0, 30.0)
```

Out[ ]:

```
(0.0, 30.0)
```

In [ ]:

```
yres_tv = test['sales']- y_pred_tv
yres_radio = test['sales']- y_pred_radio
yres_news = test['sales']- y_pred_news


print(np.mean(test['sales']- y_pred_tv))
print(np.mean(test['sales']- y_pred_radio))
print(np.mean(test['sales']- y_pred_news))
#print(y_pred_radio - test['sales'])
#print(y_pred_news - test['sales'])
```

```
-0.2354075497628375
-0.16599838561691177
0.14134471898441184
```

Residual value can be seen as a parameter of best fit of a model,it can be seen as a measure of correlation between the functions.As we know that the best model will have the mean residual close to zero.Here we observe that the model for TV might have some overfiting issues while the model used to fit the newspaper is defnitely underfiting the data.

In [ ]:

```
fig = plt.figure(figsize=(15,4))
plt.subplot(1,3,1)
ax= sns.scatterplot(data=test, x="sales", y= yres_tv);
ax.set_xlabel('TV residual')
plt.xlim(3,30)
plt.ylim(-15,15)


plt.subplot(1,3,2)
ax=sns.scatterplot(data=test, x="sales", y= yres_radio);
ax.set_xlabel('Radio residual')
plt.xlim(3,30)
plt.ylim(-15,15)


plt.subplot(1,3,3)
ax=sns.scatterplot(data=test, x="sales", y= yres_news);
ax.set_xlabel('News residual')
plt.xlim(3,30)
plt.ylim(-15,15)
```

Out[ ]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f7bd088ac18>

Out[ ]:

Text(0.5, 0, 'TV residual')

Out[ ]:

(3.0, 30.0)

Out[ ]:

(-15.0, 15.0)

Out[ ]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f7bccdf34a8>

Out[ ]:

Text(0.5, 0, 'Radio residual')

Out[ ]:

(3.0, 30.0)

Out[ ]:

(-15.0, 15.0)

Out[ ]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f7bccdf34e0>

Out[ ]:

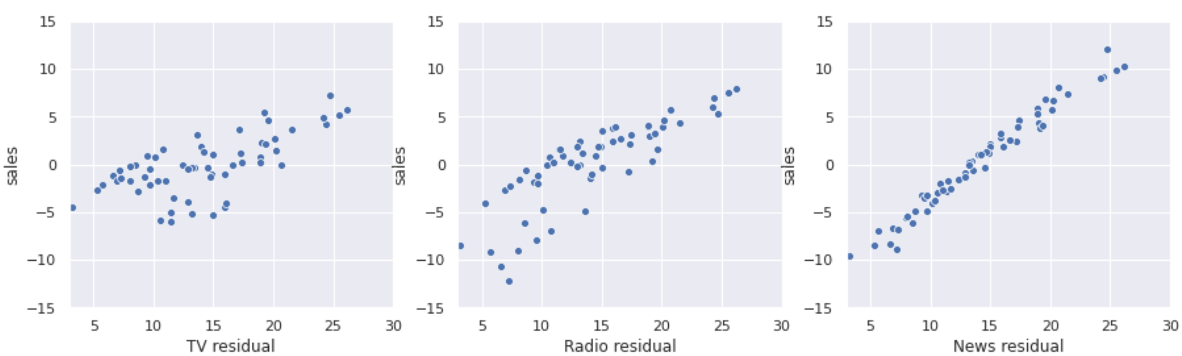
Text(0.5, 0, 'News residual')

Out[ ]:

(3.0, 30.0)

Out[ ]:

(-15.0, 15.0)



As we noticed earlier that the linear model is not the best fit for this data set so we try to various plots to get conclude that a relationship exitst between combination of various parameters. This can be achieved by plotting a scatterplot against the the features like the radio,tv and newspaper ad budget and the residual of each of the predicted models. We can observe that there exists a stong correlaton between the combination of radio and tv ad budgets and even maybe between the combination of newspaper and radio ad budgets.

In [ ]:

```
fig = plt.figure(figsize=(20,8))
plt.subplot(3,3,1)
ax= sns.scatterplot(data=test, x="TV", y= yres_tv,color='green');
ax.set_xlabel('TV residual')
ax.set_ylabel('TV budget')
ax.set_title('linear regression of tv_reg')



plt.subplot(3,3,2)
ax=sns.scatterplot(data=test, x="radio", y= yres_tv,color='green');
ax.set_xlabel('TV residual')
ax.set_ylabel('radio budget')
ax.set_title('linear regression of tv_reg')





plt.subplot(3,3,3)

ax=sns.scatterplot(data=test, x="newspaper", y= yres_tv,color='green');
ax.set_xlabel('TV residual')
ax.set_ylabel('newspaper budget')
ax.set_title('linear regression of tv_reg')



plt.subplot(3,3,4)

ax=sns.scatterplot(data=test, x="TV", y= yres_radio,color='green');
ax.set_xlabel('Radio residual')
ax.set_ylabel('TV budget')
ax.set_title('linear regression of radio_reg')



plt.subplot(3,3,5)

ax=sns.scatterplot(data=test, x="radio", y= yres_radio,color='green');
ax.set_xlabel('Radio residual')
ax.set_ylabel('radio budget')
ax.set_title('linear regression of radio_reg')



plt.subplot(3,3,6)

ax=sns.scatterplot(data=test, x="newspaper", y= yres_radio,color='green');
ax.set_xlabel('Radio residual')
ax.set_ylabel('newspaper budget')
ax.set_title('linear regression of radio_reg')
```

```
plt.subplot(3,3,7)

ax=sns.scatterplot(data=test, x="TV", y= yres_news,color='green');
ax.set_xlabel('newspaper residual')
ax.set_ylabel('TV budget')
ax.set_title('linear regression of newspaper_reg')



plt.subplot(3,3,8)

ax=sns.scatterplot(data=test, x="radio", y= yres_news,color='green');
ax.set_xlabel('newspaper residual')
ax.set_ylabel('radio budget')
ax.set_title('linear regression of newspaper_reg')

plt.subplot(3,3,9)

ax=sns.scatterplot(data=test, x="newspaper", y= yres_news,color='green');
ax.set_xlabel('newspaper residual')
ax.set_ylabel('newspaper budget')
ax.set_title('linear regression of newspaper_reg')
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7bd10685c0>
```

Out[ ]:

```
Text(0.5, 0, 'TV residual')
```

Out[ ]:

```
Text(0, 0.5, 'TV budget')
```

Out[ ]:

```
Text(0.5, 1.0, 'linear regression of tv_reg')
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7bd1068f98>
```

Out[ ]:

```
Text(0.5, 0, 'TV residual')
```

Out[ ]:

```
Text(0, 0.5, 'radio budget')
```

Out[ ]:

```
Text(0.5, 1.0, 'linear regression of tv_reg')
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7bcd5880b8>
```

Out[ ]:

```
Text(0.5, 0, 'TV residual')
```

Out[ ]:

```
Text(0, 0.5, 'newspaper budget')
```

Out[ ]:

```
Text(0.5, 1.0, 'linear regression of tv_reg')
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7bcd588828>
```

Out[ ]:

```
Text(0.5, 0, 'Radio residual')
```

Out[ ]:

```
Text(0, 0.5, 'TV budget')
```

Out[ ]:

```
Text(0.5, 1.0, 'linear regression of radio_reg')
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7bccc4c208>
```

Out[ ]:

```
Text(0.5, 0, 'Radio residual')
```

Out[ ]:

```
Text(0, 0.5, 'radio budget')
```

Out[ ]:

```
Text(0.5, 1.0, 'linear regression of radio_reg')
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7bccd1dc88>
```

Out[ ]:

```
Text(0.5, 0, 'Radio residual')
```

Out[ ]:

```
Text(0, 0.5, 'newspaper budget')
```

Out[ ]:

```
Text(0.5, 1.0, 'linear regression of radio_reg')
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7bcce9a9b0>
```

Out[ ]:

```
Text(0.5, 0, 'newspaper residual')
```

Out[ ]:

```
Text(0, 0.5, 'TV budget')
```

Out[ ]:

```
Text(0.5, 1.0, 'linear regression of newspaper_reg')
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7bcd12bac8>
```

Out[ ]:

```
Text(0.5, 0, 'newspaper residual')
```

Out[ ]:

```
Text(0, 0.5, 'radio budget')
```

Out[ ]:

```
Text(0.5, 1.0, 'linear regression of newspaper_reg')
```

Out[ ]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f7bcd0e49e8>

Out[ ]:

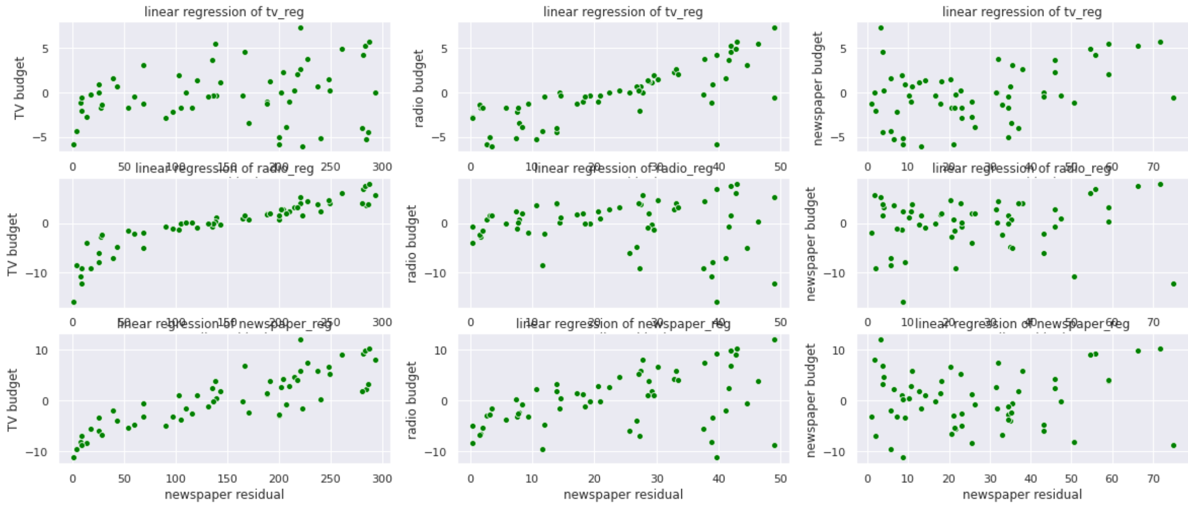
Text(0.5, 0, 'newspaper residual')

Out[ ]:

Text(0, 0.5, 'newspaper budget')

Out[ ]:

Text(0.5, 1.0, 'linear regression of newspaper_reg')



Now because of the shortcomings of the linear regression model to justify every feature we can use the mutiple regression model which analyses a combination of parameters like tv,radio and newspaper

In [ ]:

```
reg_multi_ad = LinearRegression().fit(train[['TV', 'radio', 'newspaper']], train[
'sales'])
y_train_multi_ad = reg_multi_ad.predict(train[['TV', 'radio', 'newspaper']])
r2_train_multi_ad = 1-np.mean( (y_train_multi_ad - train['sales'])**2 / np.std(tra
in['sales'])**2 )
print("Coefficients (TV, radio, newspaper):", reg_multi_ad.coef_)
print("Intercept: ", reg_multi_ad.intercept_)
print("Multiple regression: ", r2_train_multi_ad)
print("Mse for training set: ", metrics.mean_squared_error(train['sales'], y_train
_multi_ad))
```

```
Coefficients (TV, radio, newspaper): [ 0.0439533   0.20371999 -0.01021
803]
Intercept:  3.2285807385476364
Multiple regression:  0.9165077745316245
Mse for training set:  2.095096781885887
```

In [ ]:

```
y_pred_multi_ad = reg_multi_ad.predict(test[['TV', 'radio', 'newspaper']])
r2_multi_ad = 1-np.mean( (y_pred_multi_ad - test['sales'])**2 / np.std(test['sale
s'])**2 )
print("Multiple regression: ", r2_multi_ad)
print("Mse for test set: ", metrics.mean_squared_error(test['sales'], y_pred_multi
_ad))
```

```
Multiple regression:  0.851923450168499
Mse for test set:  4.69722837372494
```

In [ ]:

```
yres_multi_test = test['sales']- y_pred_multi_ad
yres_multi_train= train['sales']-y_train_multi_ad


print(np.mean(yres_multi_test))
print(np.mean(yres_multi_train))
#print(np.mean(test['sales']- y_pred_news))
#print(y_pred_radio - test['sales'])
#print(y_pred_news - test['sales'])
```

```
-0.32074614314709377
7.549516567451064e-16
```

In [ ]:

```
fig = plt.figure(figsize=(15,4))
ax=sns.scatterplot(data=train, x="sales", y= yres_multi_train,color='green');
ax= sns.scatterplot(data=test, x="sales", y= yres_multi_test);
plt.ylabel('residual')
```

Out[ ]:

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

In [ ]:

```
fig = plt.figure(figsize=(15,3))
plt.subplot(1,3,1)
ax=sns.scatterplot(data=train, x="TV", y= yres_multi_train,color='green');
ax= sns.scatterplot(data=test, x="TV", y= yres_multi_test);
ax.set_xlabel('Mutliple regression residual')
ax.set_ylabel('TV budget')
ax.set_title('multiple regression model')



plt.subplot(1,3,2)
ax=sns.scatterplot(data=train, x="radio", y= yres_multi_train,color='green');
ax= sns.scatterplot(data=test, x="radio", y= yres_multi_test);
ax.set_xlabel('Mutliple regression residual')
ax.set_ylabel('radio budget')
ax.set_title('multiple regression model')


plt.subplot(1,3,3)
ax=sns.scatterplot(data=train, x="newspaper", y= yres_multi_train,color='green');
ax= sns.scatterplot(data=test, x="newspaper", y= yres_multi_test);
ax.set_xlabel('Mutliple regression residual')
ax.set_ylabel('newspaper budget')
ax.set_title('multiple regression model')
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7bccf15780>
```

Out[ ]:

```
Text(0.5, 0, 'Mutliple regression residual')
```

Out[ ]:

```
Text(0, 0.5, 'TV budget')
```

Out[ ]:

```
Text(0.5, 1.0, 'multiple regression model')
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7bccf03e10>
```

Out[ ]:

```
Text(0.5, 0, 'Mutliple regression residual')
```

Out[ ]:

```
Text(0, 0.5, 'radio budget')
```

Out[ ]:

```
Text(0.5, 1.0, 'multiple regression model')
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7bccf50fd0>
```

Out[ ]:

```
Text(0.5, 0, 'Mutliple regression residual')
```

Out[ ]:

```
Text(0, 0.5, 'newspaper budget')
```

Out[ ]:

```
Text(0.5, 1.0, 'multiple regression model')
```

In [ ]:

```
ds['news_radio'] = ds['newspaper']*ds['radio']
ds['tv_radio'] = ds['TV']*ds['radio']
ds['news_tv'] = ds['newspaper']*ds['TV']
ds['news_radio_tv'] = ds['newspaper']*ds['radio']*ds['TV']
ds
```

Out[ ]:

	TV	radio	newspaper	sales	news_radio	tv_radio	news_tv	news_radio_tv
1	230.1	37.8	69.2	22.1	2615.76	8697.78	15922.92	601886.376
2	44.5	39.3	45.1	10.4	1772.43	1748.85	2006.95	78873.135
3	17.2	45.9	69.3	9.3	3180.87	789.48	1191.96	54710.964
4	151.5	41.3	58.5	18.5	2416.05	6256.95	8862.75	366031.575
5	180.8	10.8	58.4	12.9	630.72	1952.64	10558.72	114034.176
...	...	...	...	...	...	...	...	...
196	38.2	3.7	13.8	7.6	51.06	141.34	527.16	1950.492
197	94.2	4.9	8.1	9.7	39.69	461.58	763.02	3738.798
198	177.0	9.3	6.4	12.8	59.52	1646.10	1132.80	10535.040
199	283.6	42.0	66.2	25.5	2780.40	11911.20	18774.32	788521.440
200	232.1	8.6	8.7	13.4	74.82	1996.06	2019.27	17365.722

200 rows × 8 columns

In [ ]:

```
train, test = train_test_split(ds, test_size=0.3)
```

In [ ]:

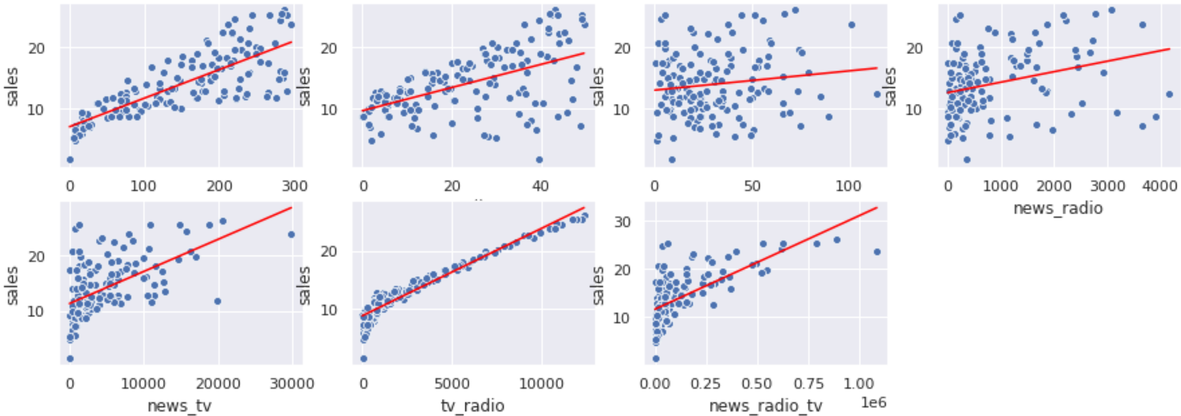
```
reg_tv = LinearRegression().fit(train[['TV']], train['sales'])
reg_radio = LinearRegression().fit(train[['radio']], train['sales'])
reg_news = LinearRegression().fit(train[['newspaper']], train['sales'])
reg_news_radio = LinearRegression().fit(train[['news_radio']], train['sales'])
reg_tv_radio = LinearRegression().fit(train[['tv_radio']], train['sales'])
reg_news_tv = LinearRegression().fit(train[['news_tv']], train['sales'])
reg_news_radio_tv = LinearRegression().fit(train[['news_radio_tv']], train['sales'
])
```

In [ ]:

```
y_pred_tv = reg_tv.predict(test[['TV']])
y_pred_radio = reg_radio.predict(test[['radio']])
y_pred_news = reg_news.predict(test[['newspaper']])
y_pred_news_radio = reg_news_radio.predict(test[['news_radio']])
y_pred_tv_radio = reg_tv_radio.predict(test[['tv_radio']])
y_pred_news_tv = reg_news_tv.predict(test[['news_tv']])
y_pred_news_radio_tv = reg_news_radio_tv.predict(test[['news_radio_tv']])
```

In [ ]:

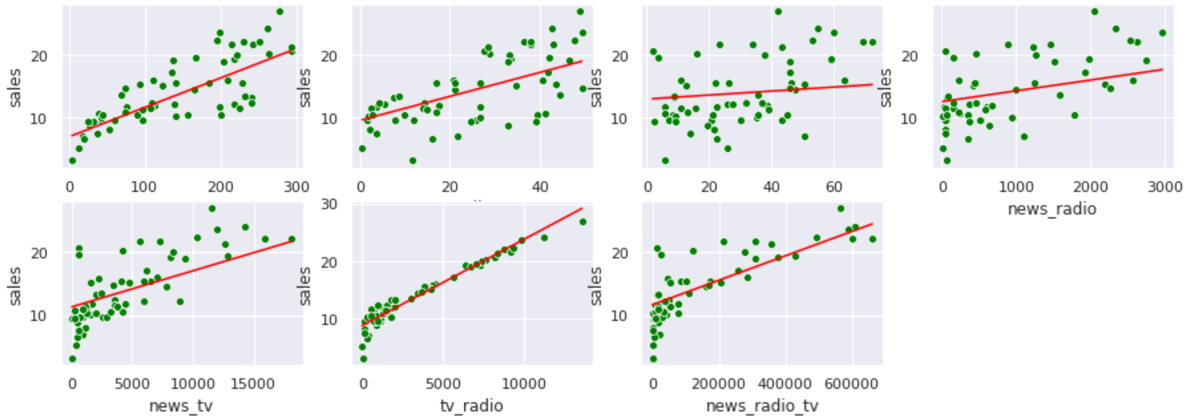
```
fig = plt.figure(figsize=(15,5))

plt.subplot(2,4,1)
sns.scatterplot(data=train, x="TV", y="sales");
sns.lineplot(data=train, x="TV", y=reg_tv.predict(train[['TV']]), color='red');

plt.subplot(2,4,2)
sns.scatterplot(data=train, x="radio", y="sales");
sns.lineplot(data=train, x="radio", y=reg_radio.predict(train[['radio']]), color=
'red');

plt.subplot(2,4,3)
sns.scatterplot(data=train,x="newspaper", y="sales");
sns.lineplot(data=train, x="newspaper", y=reg_news.predict(train[['newspaper']]),
color='red');

plt.subplot(2,4,4)
sns.scatterplot(data=train,x="news_radio", y="sales");
sns.lineplot(data=train, x="news_radio", y=reg_news_radio.predict(train[['news_rad
io']]), color='red');


plt.subplot(2,4,5)
sns.scatterplot(data=train,x="news_tv", y="sales");
sns.lineplot(data=train, x="news_tv", y=reg_news_tv.predict(train[['news_tv']]), c
olor='red');

plt.subplot(2,4,6)
sns.scatterplot(data=train,x="tv_radio", y="sales");
sns.lineplot(data=train, x="tv_radio", y=reg_tv_radio.predict(train[['tv_radio'
]]), color='red');


plt.subplot(2,4,7)
sns.scatterplot(data=train,x="news_radio_tv", y="sales");
sns.lineplot(data=train, x="news_radio_tv", y=reg_news_radio_tv.predict(train[['ne
ws_radio_tv']]), color='red');
```



After trying out various combination of features we see a strong correlaton sales and the TV*radio pair and even the news_radio_tv which shows that there is rise in sales for the overall set if we increase the budget of advertising of all the features which is a very important relation established by these graphs as we can increase the overall sales and if we want to focus on the best way to increase sales we can say that if we spend more on advertising for TV and radio we are bound to achieve maximum sales.

In [ ]:

```
fig = plt.figure(figsize=(15,5))


plt.subplot(2,4,1)
sns.scatterplot(data=test, x="TV", y="sales",color='green');
sns.lineplot(data=test, x="TV", y=reg_tv.predict(test[['TV']]), color='red');

plt.subplot(2,4,2)
sns.scatterplot(data=test, x="radio", y="sales",color='green');
sns.lineplot(data=test, x="radio", y=reg_radio.predict(test[['radio']]), color='red');

plt.subplot(2,4,3)
sns.scatterplot(data=test,x="newspaper", y="sales",color='green');
sns.lineplot(data=test, x="newspaper", y=reg_news.predict(test[['newspaper']]), color='red');

plt.subplot(2,4,4)
sns.scatterplot(data=test,x="news_radio", y="sales",color='green');
sns.lineplot(data=test, x="news_radio", y=reg_news_radio.predict(test[['news_radio']]), color='red');


plt.subplot(2,4,5)
sns.scatterplot(data=test,x="news_tv", y="sales",color='green');
sns.lineplot(data=test, x="news_tv", y=reg_news_tv.predict(test[['news_tv']]), color='red');

plt.subplot(2,4,6)
sns.scatterplot(data=test,x="tv_radio", y="sales",color='green');
sns.lineplot(data=test, x="tv_radio", y=reg_tv_radio.predict(test[['tv_radio']]), color='red');


plt.subplot(2,4,7)
sns.scatterplot(data=test,x="news_radio_tv", y="sales",color='green');
sns.lineplot(data=test, x="news_radio_tv", y=reg_news_radio_tv.predict(test[['news_radio_tv']]), color='red');
```



As discussed earlier the closer the mean residual value to zero, the better is the model. This point is clarified below as we notice that the linear model proves to be quite efficient to describe the sales of combination of TV and radio(TV*Radio).

In [ ]:

```
yres_tv = test['sales']- y_pred_tv
yres_radio = test['sales']- y_pred_radio
yres_news = test['sales']- y_pred_news
yres_news_radio = test['sales']- y_pred_news_radio
yres_news_tv= test['sales']- y_pred_news_tv
yres_tv_radio = test['sales']- y_pred_tv_radio
yres_news_radio_tv = test['sales']- y_pred_news_radio_tv
print(np.mean(yres_tv))
print(np.mean(yres_radio))
print(np.mean(yres_news))
print(np.mean(yres_news_radio))
print(np.mean(yres_news_tv))
print(np.mean(yres_tv_radio))
print(np.mean(yres_news_radio_tv))
```

```
0.4994855309210653
-0.09728225329292188
0.17751457429743606
0.051031355268480814
0.21214840234654073
-0.05657004822411074
-0.2371146202107957
```

In [ ]:

```
fig = plt.figure(figsize=(15,5))
plt.subplot(2,4,1)
ax= sns.scatterplot(data=test, x="sales", y= yres_tv);
ax.set_xlabel('TV residual')


plt.subplot(2,4,2)
ax=sns.scatterplot(data=test, x="sales", y= yres_radio);
ax.set_xlabel('Radio residual')


plt.subplot(2,4,3)
ax=sns.scatterplot(data=test, x="sales", y= yres_news);
ax.set_xlabel('News residual')


plt.subplot(2,4,4)
ax= sns.scatterplot(data=test, x="sales", y= yres_news_radio);
ax.set_xlabel('news_radio residual')


plt.subplot(2,4,5)
ax=sns.scatterplot(data=test, x="sales", y= yres_news_tv);
ax.set_xlabel('news TV residual')


plt.subplot(2,4,6)
ax=sns.scatterplot(data=test, x="sales", y=yres_tv_radio);
ax.set_xlabel('TV radio residual')

plt.subplot(2,4,7)
ax=sns.scatterplot(data=test, x="sales", y=yres_news_radio_tv);
ax.set_xlabel('news radio tv residual')
```

Out[ ]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f7bcc7f1c18>

Out[ ]:

Text(0.5, 0, 'TV residual')

Out[ ]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f7bcc7f1c88>

Out[ ]:

Text(0.5, 0, 'Radio residual')

Out[ ]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f7bcc7f17b8>

Out[ ]:

Text(0.5, 0, 'News residual')

Out[ ]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f7bcc330f60>

Out[ ]:

Text(0.5, 0, 'news_radio residual')

Out[ ]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f7bcc3240b8>

Out[ ]:

Text(0.5, 0, 'news TV residual')

Out[ ]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f7bcc2f2c50>

Out[ ]:

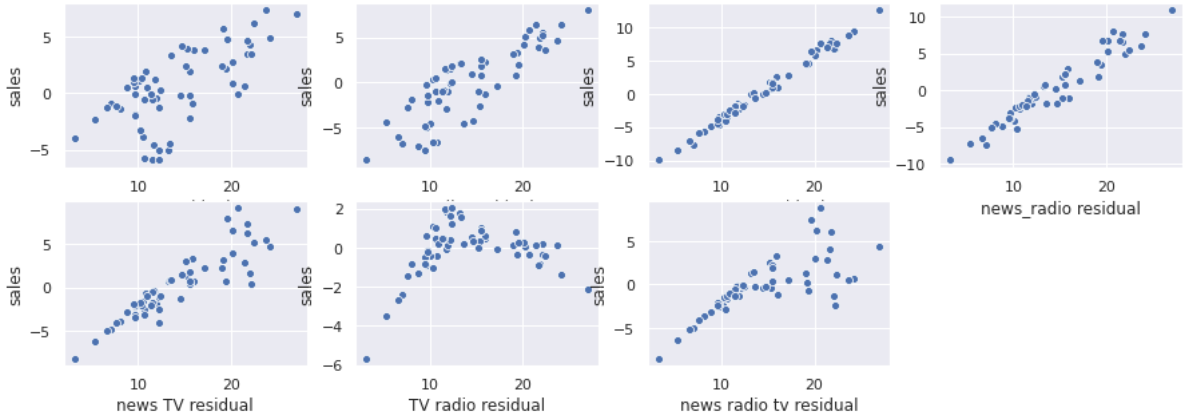
Text(0.5, 0, 'TV radio residual')

Out[ ]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f7bcc2a14e0>

Out[ ]:

Text(0.5, 0, 'news radio tv residual')



This is just a quick way to analyse what we did till now using statsmodels library,which gives us all the important relation parameters and gives us the R2 score the accuracy and the interdependece of the different features. The Condition Number here tells us that there is a very high correlation between the different features selected by us, in our case it can be the feature of radio*TV.

In [ ]:

```
import statsmodels.formula.api as sm
model1 = sm.ols(formula="sales~radio*newspaper", data=ds).fit()
print(model1.summary())
model2 = sm.ols(formula="sales~TV*newspaper", data=ds).fit()
print(model2.summary())
model3 = sm.ols(formula="sales~radio*TV", data=ds).fit()
print(model3.summary())
model4 = sm.ols(formula="sales~radio*newspaper*TV", data=ds).fit()
print(model4.summary())
```

```
                            OLS Regression Results
==============================================================================
========
Dep. Variable:                  sales   R-squared:
0.334
Model:                            OLS   Adj. R-squared:
0.323
Method:                 Least Squares   F-statistic:
32.70
Date:                Fri, 26 Jun 2020   Prob (F-statistic):
3.51e-17
Time:                        06:42:21   Log-Likelihood:
-573.11
No. Observations:                 200   AIC:
1154.
Df Residuals:                     196   BIC:
1167.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
=============
                      coef    std err          t      P>|t|      [0.02
5      0.975]
------------------------------------------------------------------------------
-------------
Intercept           8.7905      1.022      8.597      0.000       6.77
4      10.807
radio               0.2146      0.038      5.603      0.000       0.13
9       0.290
newspaper           0.0221      0.035      0.638      0.524      -0.04
6       0.090
radio:newspaper    -0.0005      0.001     -0.494      0.622      -0.00
3       0.002
==============================================================================
========
Omnibus:                       18.936   Durbin-Watson:
1.934
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
21.320
Skew:                          -0.756   Prob(JB):
2.35e-05
Kurtosis:                       3.524   Cond. No.
4.21e+03
==============================================================================
========

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 4.21e+03. This might indicate that
there are
strong multicollinearity or other numerical problems.
                            OLS Regression Results
==============================================================================
========
Dep. Variable:                  sales   R-squared:
0.649
```

```
Model:                            OLS   Adj. R-squared:
0.643
Method:                 Least Squares   F-statistic:
120.6
Date:                Fri, 26 Jun 2020   Prob (F-statistic):
2.84e-44
Time:                        06:42:21   Log-Likelihood:
-509.12
No. Observations:                 200   AIC:
1026.
Df Residuals:                     196   BIC:
1039.
Df Model:                           3
Covariance Type:            nonrobust
======================================================================
==========
                   coef    std err          t      P>|t|      [0.025
0.975]
----------------------------------------------------------------------
----------
Intercept        6.4042      0.733      8.732      0.000       4.958
7.851
TV               0.0427      0.004      9.896      0.000       0.034
0.051
newspaper        0.0241      0.019      1.251      0.212      -0.014
0.062
TV:newspaper     0.0001      0.000      1.228      0.221   -8.03e-05
0.000
======================================================================
========
Omnibus:                        1.120   Durbin-Watson:
1.940
Prob(Omnibus):                  0.571   Jarque-Bera (JB):
0.778
Skew:                          -0.084   Prob(JB):
0.678
Kurtosis:                       3.256   Cond. No.
2.23e+04
======================================================================
========

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 2.23e+04. This might indicate that
there are
strong multicollinearity or other numerical problems.
                            OLS Regression Results
======================================================================
========
Dep. Variable:                  sales   R-squared:
0.968
Model:                            OLS   Adj. R-squared:
0.967
Method:                 Least Squares   F-statistic:
1963.
Date:                Fri, 26 Jun 2020   Prob (F-statistic):
```

```
6.68e-146
Time:                        06:42:21   Log-Likelihood:
-270.14
No. Observations:                 200   AIC:
548.3
Df Residuals:                     196   BIC:
561.5
Df Model:                           3
Covariance Type:            nonrobust
======================================================================
========
                 coef    std err          t      P>|t|      [0.025
0.975]
----------------------------------------------------------------------
--------
Intercept      6.7502      0.248     27.233      0.000       6.261
7.239
radio          0.0289      0.009      3.241      0.001       0.011
0.046
TV             0.0191      0.002     12.699      0.000       0.016
0.022
radio:TV       0.0011   5.24e-05     20.727      0.000       0.001
0.001
======================================================================
========
Omnibus:                      128.132   Durbin-Watson:
2.224
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
1183.719
Skew:                          -2.323   Prob(JB):
9.09e-258
Kurtosis:                      13.975   Cond. No.
1.80e+04
======================================================================
========

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 1.8e+04. This might indicate that t
here are
strong multicollinearity or other numerical problems.
                            OLS Regression Results
======================================================================
========
Dep. Variable:                  sales   R-squared:
0.969
Model:                            OLS   Adj. R-squared:
0.968
Method:                 Least Squares   F-statistic:
847.3
Date:                Fri, 26 Jun 2020   Prob (F-statistic):
1.16e-140
Time:                        06:42:21   Log-Likelihood:
-267.45
No. Observations:                 200   AIC:
550.9
```

```
Df Residuals:                     192   BIC:
577.3
Df Model:                           7
Covariance Type:            nonrobust
======================================================================
================
                         coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------
----------------
Intercept              6.5559      0.466     14.083      0.000
5.638       7.474
radio                  0.0196      0.016      1.197      0.233      -
0.013       0.052
newspaper              0.0131      0.017      0.761      0.447      -
0.021       0.047
radio:newspaper     9.063e-06      0.000      0.019      0.985      -
0.001       0.001
TV                     0.0197      0.003      7.250      0.000
0.014       0.025
radio:TV               0.0012   9.75e-05     11.909      0.000
0.001       0.001
newspaper:TV       -5.546e-05   9.33e-05     -0.595      0.553      -
0.000       0.000
radio:newspaper:TV  -7.61e-07    2.7e-06     -0.282      0.778   -6.09
e-06    4.57e-06
======================================================================
========
Omnibus:                      110.676   Durbin-Watson:
2.224
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
751.534
Skew:                          -2.035   Prob(JB):
6.40e-164
Kurtosis:                      11.580   Cond. No.
1.56e+06
======================================================================
========

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 1.56e+06. This might indicate that
there are
strong multicollinearity or other numerical problems.
```