In [ ]:

Project: Linear Regression analysis on "advertising dataset" I have downloaded the "Advertising.csv" file from "An Introduction to Statistical Learning with Applications in R". The aim of this project is to show my skills in data analysis and linear regression modeling using Python and scikit-learn. My work is for educational purposes.

In [ ]:
```
# Python_Machine_Learning_Linear_Regression_project_01
```

In [152]:
```
# Load Pandas Library

import pandas as pd
```

In [173]:
```
# Load the ("Advertising.csv") file
df = pd.read_csv("Advertising.csv")

# Look at the first few rows.
df
```

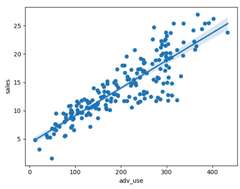
Out[173]:

	Unnamed: 0	TV	radio	newspaper	sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9
...	...	...	...	...	...
195	196	38.2	3.7	13.8	7.6
196	197	94.2	4.9	8.1	9.7
197	198	177.0	9.3	6.4	12.8
198	199	283.6	42.0	66.2	25.5
199	200	232.1	8.6	8.7	13.4

200 rows × 5 columns

In [139]:
```
# Find total advertising cost

df["adv_use"] = df["TV"] + df["radio"] + df["newspaper"]    #  "adv_use" is a new column in df dataframe

import seaborn as sns
# scatter plot between "adv_use" and "sales".
sns.scatterplot(data = df, x = "adv_use", y = "sales")

# Loads regression as the best fit line with x and y values.
sns.regplot(data = df, x = "adv_use", y = "sales")
```

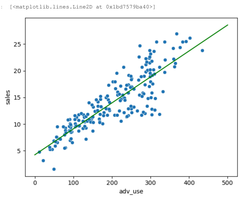
Out[139]:
```
<Axes: xlabel='adv_use', ylabel='sales'>
```



In [27]:
```
# predict the sales using a simple linear regression method (deg. = 1)

import numpy as np

x = df["adv_use"]
y = df['sales']

# Use y = mx + b with deg = 1 for x,
print(np.polyfit(x, y, deg = 1)) # it gives slope (m) and intercept (b).

# Now for any x, we can predict y, for example.

total_use = np.linspace(0, 500, 500)
expect_sales = 0.04868788 * total_use + 4.24302822

# plot
sns.scatterplot(x = "adv_use", y = 'sales', data = df)

import matplotlib.pyplot as plt
plt.plot(total_use, expect_sales, color = 'green')  # "green" line is the fitted line on data.
```

```
[0.04868788 4.24302822]
```

Out[27]:
```
[<matplotlib.lines.Line2D at 0x1bd7579ba40>]
```

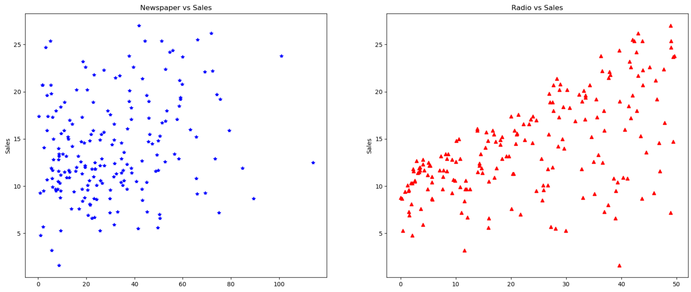


In [29]:
```
# Find sales at total advertising cost = 320. # This is an example

total_use = 320
expect_sales = 0.04868788 * total_use + 4.24302822
print(expect_sales)
```

```
19.82316982
```

In [33]:
```
# Plot  "newspaper" vs "Sales" and "radio" vs "Sales"

fig, axes = plt.subplots(nrows = 1, ncols = 2, figsize = (20, 8))

axes[0].plot(df['newspaper'], df['sales'], '*', color='blue')
axes[0].set_ylabel('Sales')
axes[0].set_title('Newspaper vs Sales')

axes[1].plot(df['radio'], df['sales'], '*', color='red')
axes[1].set_ylabel('Sales')
axes[1].set_title('Radio vs Sales')
```

Out[33]:
```
Text(0.5, 1.0, 'Radio vs Sales')
```



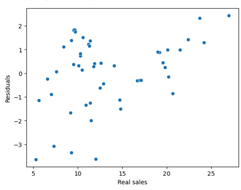
Scikit Learn used for data analysis under supervised machine learning method

In [53]:
```
y = df['sales']

X = df.drop('sales', axis = 1)

# Load train test split
from sklearn.model_selection import train_test_split

# Break data into train and test parts
# test_size = 25% of total datasets (df)
# Choose random_state = 101
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 101)

# Use linear Regression Model for training datasets
from sklearn.linear_model import LinearRegression

# Initialize the above model
model = LinearRegression()

# Fit the model with the training dataset
model.fit(X_train, y_train)

# predict the y-values for x test data using the trained model.

X_test_predict = model.predict(X_test)
print(X_test_predict)

# Check the model result
# Load mean absolute error, and mean squared error

from sklearn.metrics import mean_absolute_error, mean_squared_error

mean_absolute_error(y_test, X_test_predict)
print (mean_absolute_error(y_test, X_test_predict))

# Calculate mean squared error

mean_squared_error(y_test, X_test_predict)
print (mean_squared_error(y_test, X_test_predict))

# Calculate root mean square error
print(np.sqrt(mean_squared_error(y_test, X_test_predict)))
```

```
[15.83191242 19.55869539 11.49273205 17.00504214  9.13124181  6.84848718
 20.3544749  17.3000898   9.56481472 19.11446851 12.2239952  13.78475287
 15.44657927 21.38464974 18.3230431   9.78752457 15.62018215  7.4812015
  7.29105722 20.50885394  7.53182401 18.11190032 24.56579696 22.90995849
  7.93828626 12.65363832 21.56169469  7.87647485 12.1723918  12.64612267
 10.87969284 19.15665503  9.9720071   6.73774424 17.39318827  7.76879855
  8.09059493  7.3592284  10.39756087 10.36245547 13.12580189  9.48399068
 10.02581018  7.89544968 11.52022969 10.14232472  8.9377022  16.29813719
 13.36686143 20.98315793]
1.1714220139314558
2.1676837364568857
1.4723055847471473
```

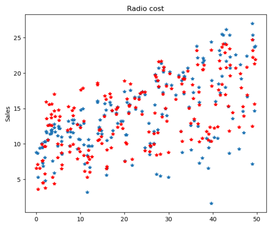
In [55]:
```
# residuals plot
X_test_residuals = y_test - X_test_predict
#print(X_test_residuals)

sns.scatterplot(x = y_test, y = X_test_residuals)  # more or less random.
plt.xlabel("Real sales")
plt.ylabel("Residuals")
```

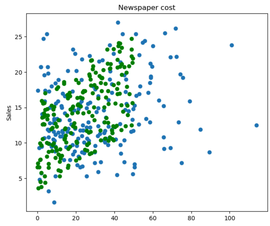
Out[55]:
```
Text(0, 0.5, 'Residuals')
```



In [57]:
```
# # create a model.
alldata_model = LinearRegression()   # All parameters here are default.

# fit all data
alldata_model.fit(X, y)

y_predict_values = alldata_model.predict(X) # predict y value from X ( TV, radio, newspaper) using the

#print(y_predict_values )

fig, axes = plt.subplots(nrows = 1, ncols = 2, figsize = (14, 5))

# First row and first column
axes[0].plot(df['radio'], df['sales'], '*')  # true point is blue
axes[0].plot(df['radio'], y_predict_values , '*', color = 'red') # predicted point is red.
axes[0].set_ylabel('Sales')
axes[0].set_title("Radio cost")

# First row and second column
axes[1].plot(df['newspaper'], df['sales'], '*')
axes[1].plot(df['radio'], y_predict_values , 'o', color = 'green')
axes[1].set_ylabel('Sales')
axes[1].set_title("Newspaper cost")
```

Out[57]:
```
Text(0.5, 1.0, 'Newspaper cost')
```


Radio cost
Sales
0
10
20
30
40
50
5
10
15
20
25


Newspaper cost
Sales
0
20
40
60
80
100
5
10
15
20
25

In [ ]:

In [ ]:

In [ ]: