Dự đoán doanh số

(Hồi quy tuyến tính cơ bản)

Vấn đề

Xây dựng một mô hình dự đoán doanh số bán hàng dựa trên số tiền chi cho các nền tảng tiếp thị khác nhau.

Dữ liệu

Sử dụng bộ dữ liệu quảng cáo được cung cấp trong ISLR và phân tích mối quan hệ giữa 'quảng cáo truyền hình' và 'doanh số bán hàng' bằng cách sử dụng mô hình hồi quy tuyến tính đơn giản.

1. Đọc và hiểu dữ liệu

In [20]:
```
# Import the numpy and pandas package

import numpy as np
import pandas as pd

# Data Visualisation
import matplotlib.pyplot as plt
import seaborn as sns
```

1.1. Đọc dữ liệu

Đọc dữ liệu từ file CSV và in ra 5 dòng đầu tiên

In [21]:
```
advertising = pd.DataFrame(pd.read_csv("advertising.csv"))
advertising.head()
```

Out[21]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

1.2. In kích thước của dữ liệu

Keyword: Shape

In [22]:
```
advertising.shape
```

Out[22]: (200, 4)

1.3. In ra thông tin của dữ liệu

Keyword: Info

In [23]:
```
advertising.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

1.4. In ra bảng thống kê mô tả

Keyword: Describe()

In [24]:
```
advertising.describe()
```

Out[24]:

	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000	36.525000	45.100000	19.050000
max	296.400000	49.600000	114.000000	27.000000

2. Làm sạch dữ liệu

2.1. Kiểm tra null

Keyword: isnull, sum

In [25]:
```
advertising.isnull().sum()
```

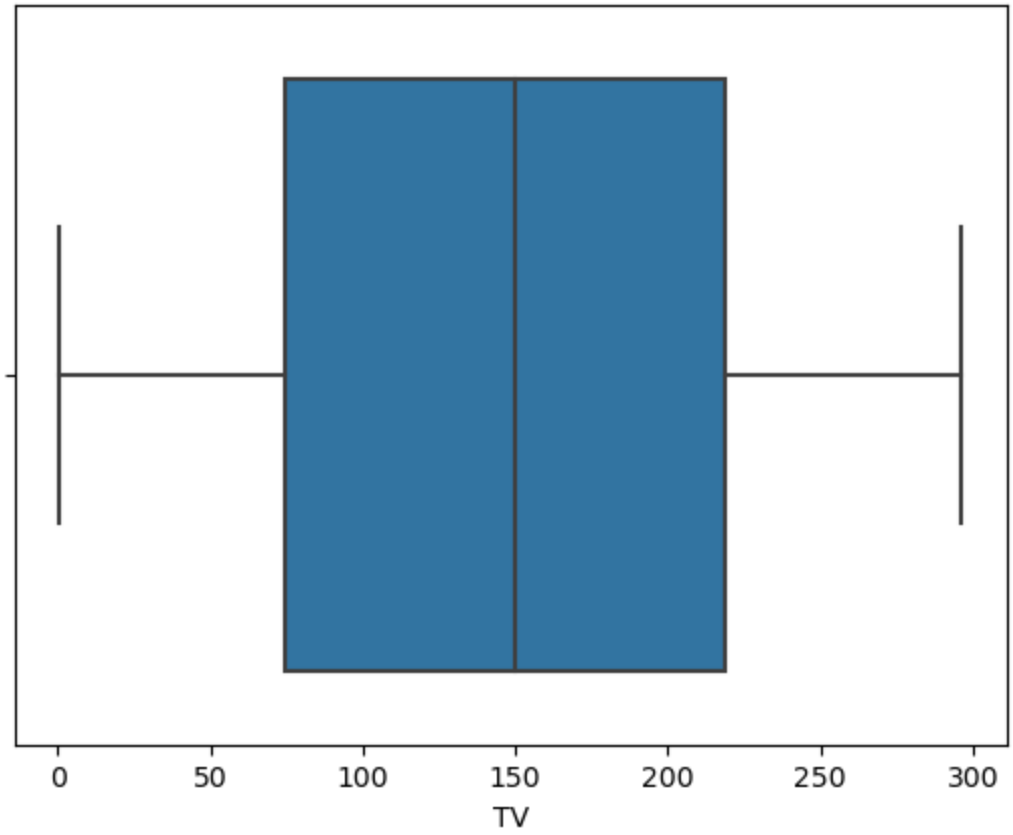
Out[25]:
```
TV          0
Radio       0
Newspaper   0
Sales       0
dtype: int64
```

2.2. Kiểm tra outlier

Vẽ boxplot cho dữ liệu TV

In [26]:
```
sns.boxplot(advertising['TV'])
plt.show()
```

```
/Users/baovu/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



3. Khám phá dữ liệu

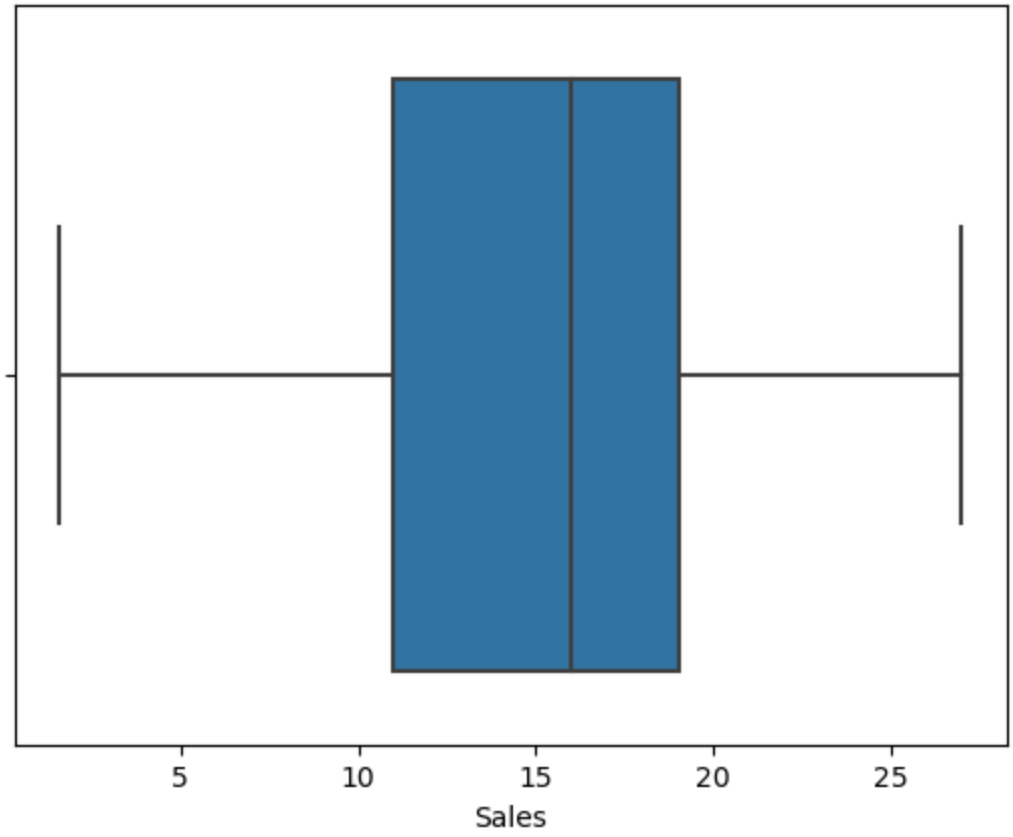
Vẽ boxplot của Sale

3.1. Vẽ boxplot cho Sale

Vẽ boxplot của Sale

In [27]:
```
sns.boxplot(advertising['Sales'])
plt.show()
```

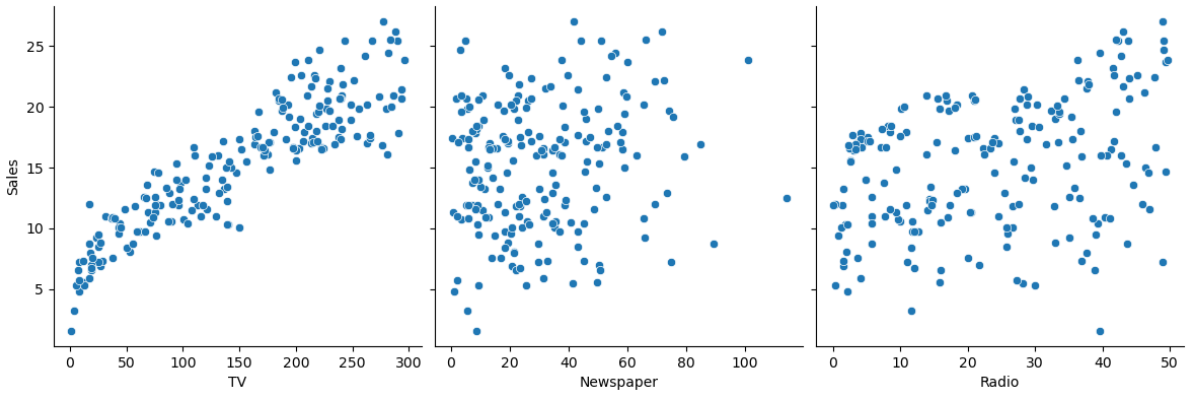
```
/Users/baovu/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



3.2. Vẽ pairplot

Vẽ pairplot để thấy tương quan giữa Sales và các biến khác.

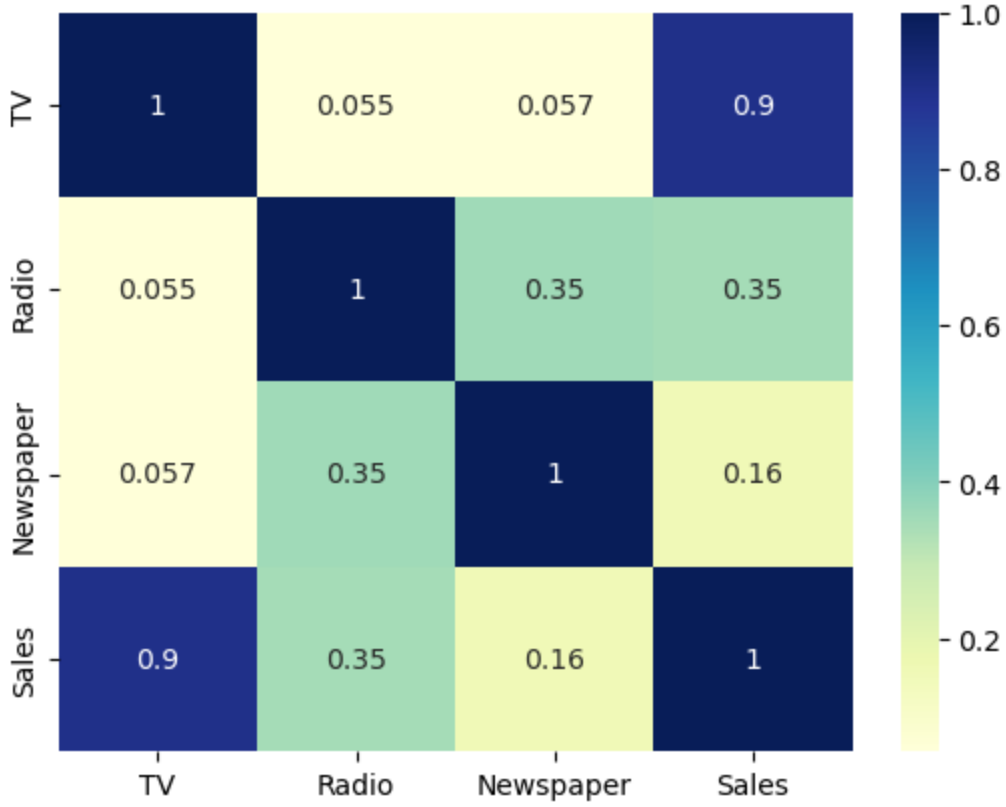
In [28]:
```
sns.pairplot(advertising, x_vars=['TV', 'Newspaper', 'Radio'], y_vars='Sales', height=4, aspect=1, kind='scatte
plt.show()
```



3.3. Vẽ heatmap

Vẽ heatmap để thấy tương quan giữa các biến dữ liệu với nhau

In [29]:
```
# Let's see the correlation between different variables.
sns.heatmap(advertising.corr(), cmap="YlGnBu", annot = True)
plt.show()
```



4. Xây dựng mô hình

Phương trình hồi quy tuyến tính y=c+m1x1+m2x2+...+mnxn

y là kết quả c là chặn m1 là hệ số cho biến đầu tiên mn là hệ số cho biến thứ n

Trong trường hợp của chúng ta:

y=c+m1×TV

m là các giá trị được gọi là hệ số mô hình hoặc tham số mô hình

4.1. Tạo biến X và Y

X lưu dữ liệu TV, Y lưu dữ liệu Sales

In [30]:
```
X = advertising['TV']
y = advertising['Sales']
print(X)
print(y)
```

```
0      230.1
1       44.5
2       17.2
3      151.5
4      180.8
       ...
195     38.2
196     94.2
197    177.0
198    283.6
199    232.1
Name: TV, Length: 200, dtype: float64
0     22.1
1     10.4
2     12.0
3     16.5
4     17.9
      ...
195    7.6
196   14.0
197   14.8
198   25.5
199   18.4
Name: Sales, Length: 200, dtype: float64
```

4.2. Chia tập train và tập test

Sử dụng sklearn.model_selection để chia bộ dữ liệu thành 2 phần, 1 phần gọi là tập huấn luyện (train), phần còn lại gọi là tập kiểm thử (test) Tập train chiếm 70% và tập test chiếm 30%

In [31]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7, test_size = 0.3, random_state = 100)
print(X_train)
print(X_test)
print(y_train)
print(y_test)
```

```
74     213.4
3      151.5
185    205.0
26     142.9
90     134.3
       ...
87     110.7
103    187.9
67     139.3
24      62.3
8        8.6
Name: TV, Length: 140, dtype: float64
126      7.8
104    238.2
99     135.2
92     217.7
111    241.7
167    206.8
116    139.2
```

4.3. Xem tập train và test

In ra 5 dòng đầu tiên của tập train

In [32]:
```
X_train.head()
```

Out[32]:
```
74     213.4
3      151.5
185    205.0
26     142.9
90     134.3
Name: TV, dtype: float64
```

In [33]:
```
y_train.head()
```

Out[33]:
```
74     17.0
3      16.5
185    22.6
26     15.0
90     14.0
Name: Sales, dtype: float64
```

4.4. Xây dựng mô hình

Sử dụng statsmodels.api để xây dựng mô hình

Khai báo thư viện

In [34]:
```
import statsmodels.api as sm
```

Theo mặc định, thư viện statsmodels phù hợp với một dòng trên tập dữ liệu đi qua điểm gốc. Nhưng để có phần chặn, bạn cần sử dụng thủ công thuộc tính add_constant của statsmodels. Và khi bạn đã thêm hằng số vào tập dữ liệu X_train của mình, bạn có thể tiếp tục và điều chỉnh đường hồi quy bằng thuộc tính OLS (Bình phương nhỏ nhất thông thường) của mô hình thống kê.

Gọi hàm add_constant để thêm chặn (intercept)

In [35]:
```
X_train_sm = sm.add_constant(X_train)
```

Gọi hàm OLS và fit để tìm ra regression line

In [36]:
```
lr = sm.OLS(y_train, X_train_sm).fit()
```

In các tham số, tức là phần chặn và độ dốc của đường hồi quy

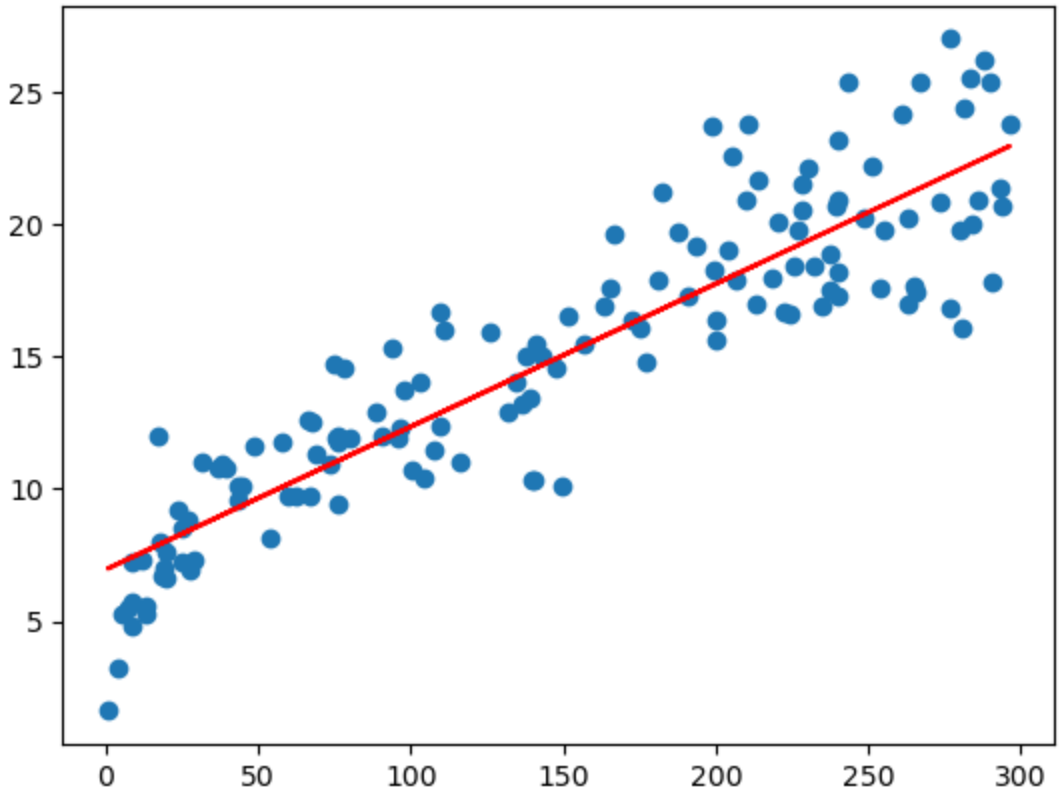
In [37]:
```
lr.params
```

Out[37]:
```
const    6.948683
TV       0.054546
dtype: float64
```

Kết quả tìm được sẽ là: Sales=6.948+0.054×TV

Vẽ biểu đồ scatter và line để thấy rõ kết quả

In [38]:
```
plt.scatter(X_train, y_train)
plt.plot(X_train, 6.948 + 0.054*X_train, 'r')
plt.show()
```



5. Đánh giá mô hình

Chúng ta cần kiểm tra xem các số hạng sai số cũng có phân phối chuẩn hay không (thực tế là một trong những giả định chính của hồi quy tuyến tính), chúng ta hãy vẽ biểu đồ của các số hạng sai số và xem nó trông như thế nào.

5.1. Đánh giá trên tập train

Gọi hàm predict trên tập train, sau đó tìm độ chênh lệch giữa kết quả dự đoán trên tập train và kết quả thực tế.

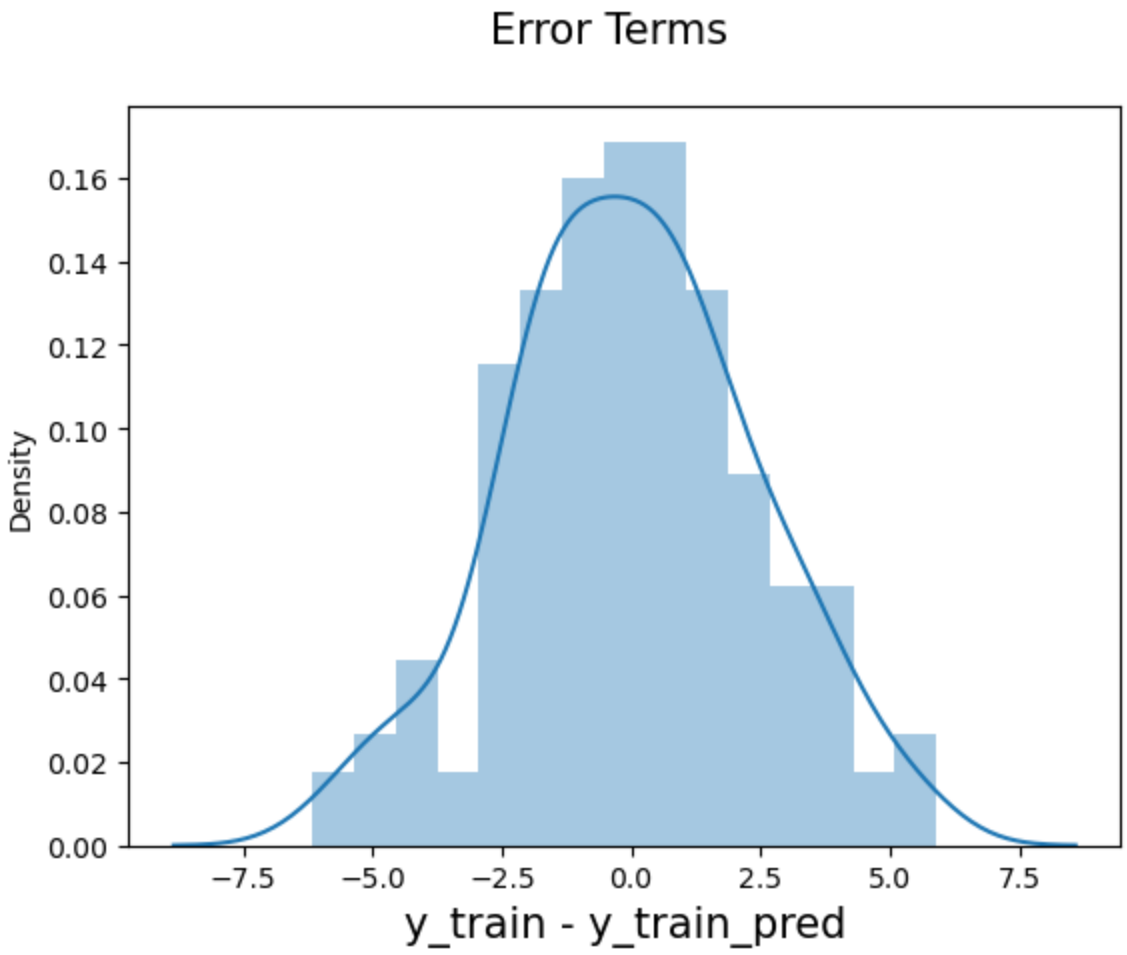
In [39]:
```
y_train_pred = lr.predict(X_train_sm)
res = (y_train - y_train_pred)
print(res)
```

```
74    -1.588747
3      1.287635
185    4.469437
26     0.256729
90    -0.274178
         ...
87     3.013102
103    2.502170
67    -1.146907
24    -0.646884
8     -2.617777
Length: 140, dtype: float64
```

Vẽ biểu đồ để thấy rõ hơn

In [40]:
```
fig = plt.figure()
sns.distplot(res, bins = 15)
fig.suptitle('Error Terms', fontsize = 15)                # Plot heading
plt.xlabel('y_train - y_train_pred', fontsize = 15)       # X-label
plt.show()
```

```
/Users/baovu/opt/anaconda3/lib/python3.9/site-packages/seaborn/distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use eit
her `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histog
rams).
  warnings.warn(msg, FutureWarning)
```



5.2. Đánh giá trên tập test

Làm tương tự trên tập train

In [41]:
```
# Add a constant to X_test
X_test_sm = sm.add_constant(X_test)

# Predict the y values corresponding to X_test_sm
y_pred = lr.predict(X_test_sm)

y_pred.head()
```

Out[41]:
```
126     7.374140
104    19.941482
99     14.323269
92     18.823294
111    20.132392
dtype: float64
```

Khai báo thư viên để dùng mean_squared_error và r2_score

In [42]:
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

Đánh giá qua mean_squared_error

In [43]:
```
np.sqrt(mean_squared_error(y_test, y_pred))
```

Out[43]: 2.019296008966233

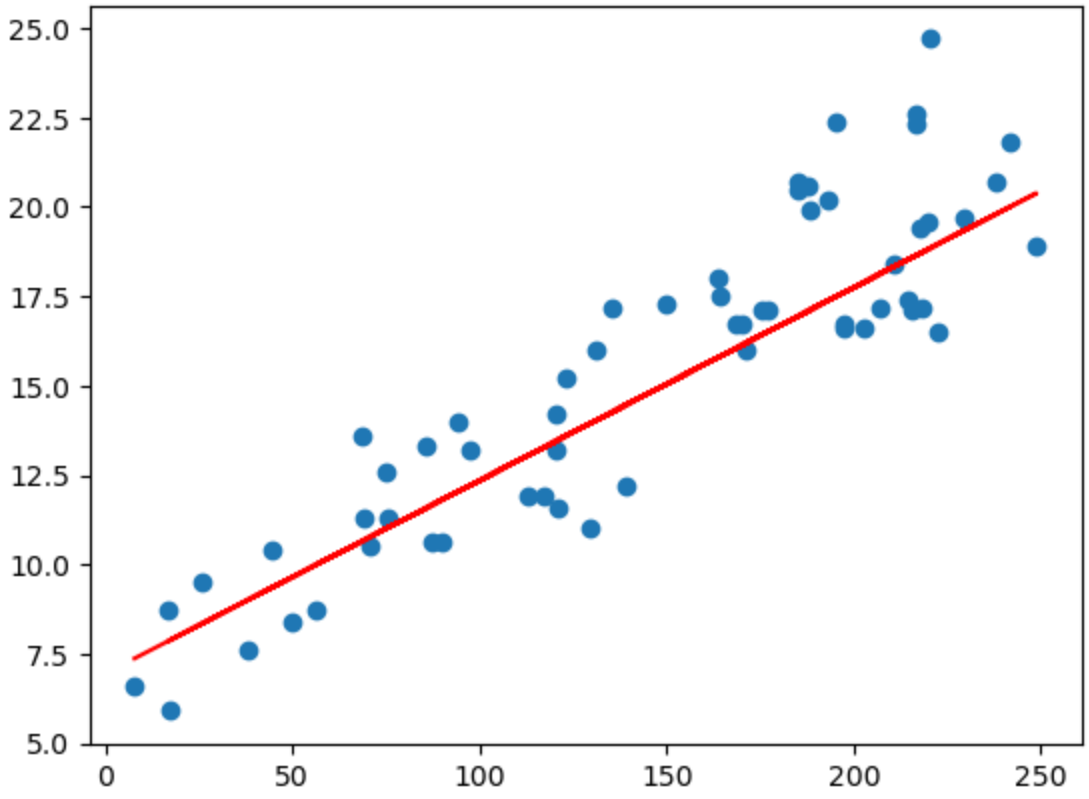
Đánh giá qua r2_score

In [44]:
```
r_squared = r2_score(y_test, y_pred)
r_squared
```

Out[44]: 0.7921031601245658

Vẽ biểu đồ để xem mô hình khớp với dữ liệu thực tế thế nào

In [45]:
```
plt.scatter(X_test, y_test)
plt.plot(X_test, 6.948 + 0.054 * X_test, 'r')
plt.show()
```



In [ ]: