In [16]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as snr
from sklearn.preprocessing import OneHotEncoder, PolynomialFeatures
from sklearn.metrics import r2_score
```

In [3]:

```
data=pd.read_csv("C:\\Users\\Lenovo\\Desktop\\CodeSoft\\Task4_SalesPrediction\\adve
data.head(10)
```

Out[3]:

	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
5	8.7	48.9	75.0	7.2
6	57.5	32.8	23.5	11.8
7	120.2	19.6	11.6	13.2
8	8.6	2.1	1.0	4.8
9	199.8	2.6	21.2	15.6

In [6]:

```
data.shape
```

Out[6]:

```
(200, 4)
```

In [7]:

```
data.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
```

In [8]:

```
data.describe()
```

Out[8]:

	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

In [10]:

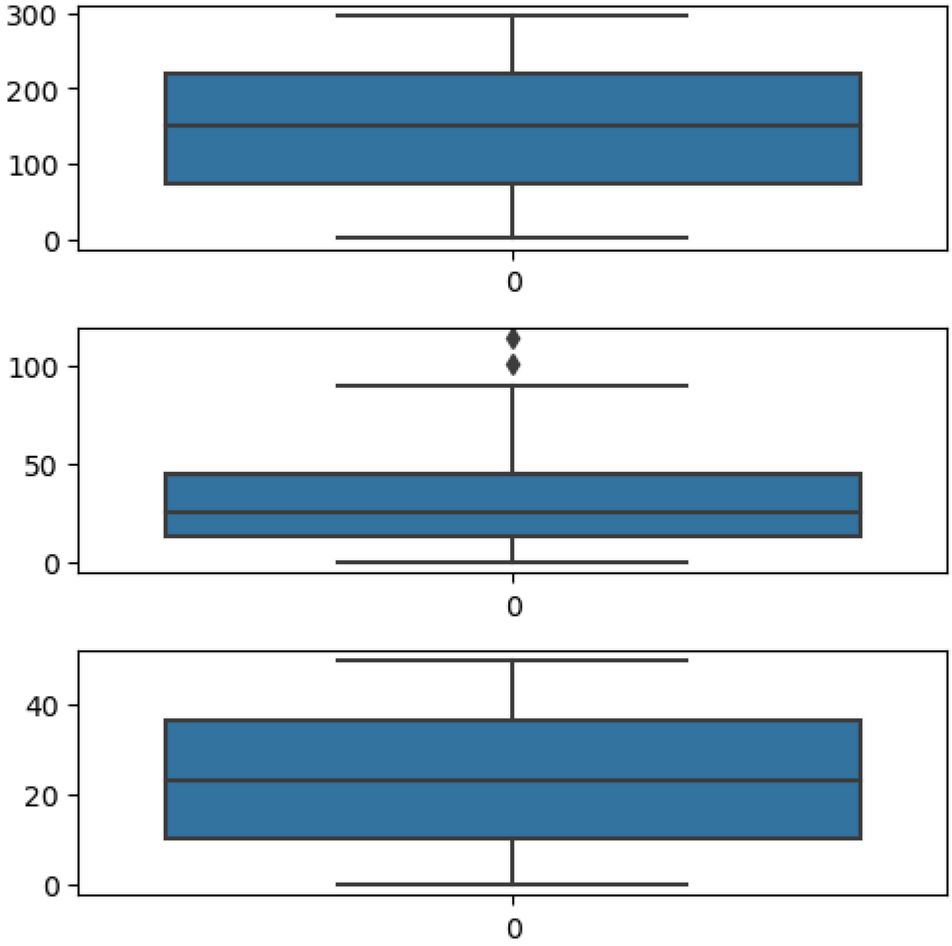
```
data.isnull().sum()*100/data.shape[0]
```

Out[10]:

```
TV           0.0
Radio        0.0
Newspaper    0.0
Sales        0.0
dtype: float64
```

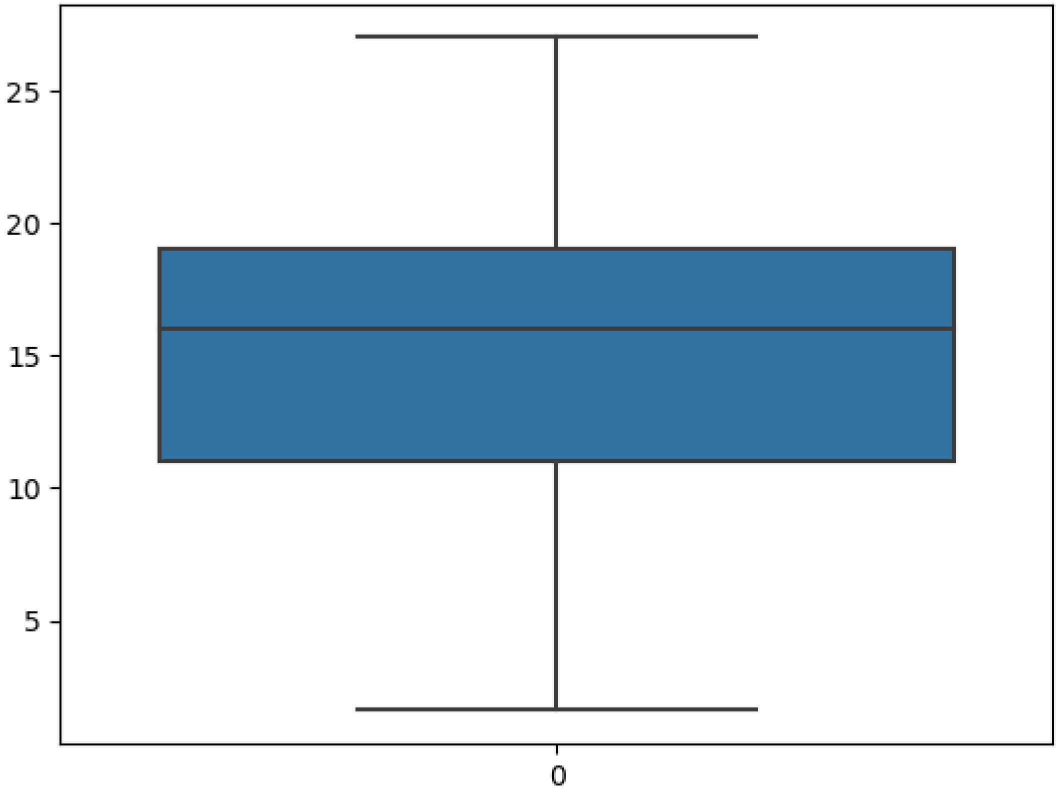
In [17]:

```
fig, axs = plt.subplots(3, figsize = (5, 5))
plt1 = snr.boxplot(data['TV'], ax = axs[0])
plt2 = snr.boxplot(data['Newspaper'], ax = axs[1])
plt3 = snr.boxplot(data['Radio'], ax = axs[2])
plt.tight_layout()
```



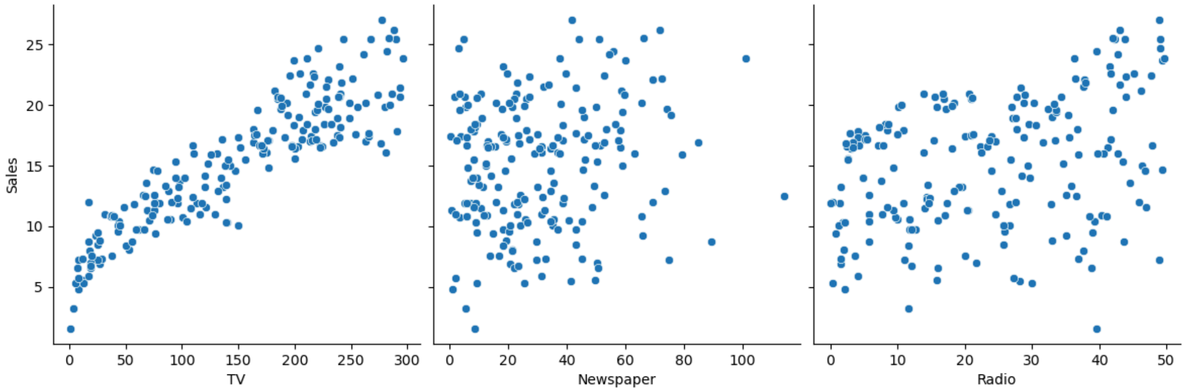
In [18]:

```
snr.boxplot(data['Sales'])
plt.show()
```

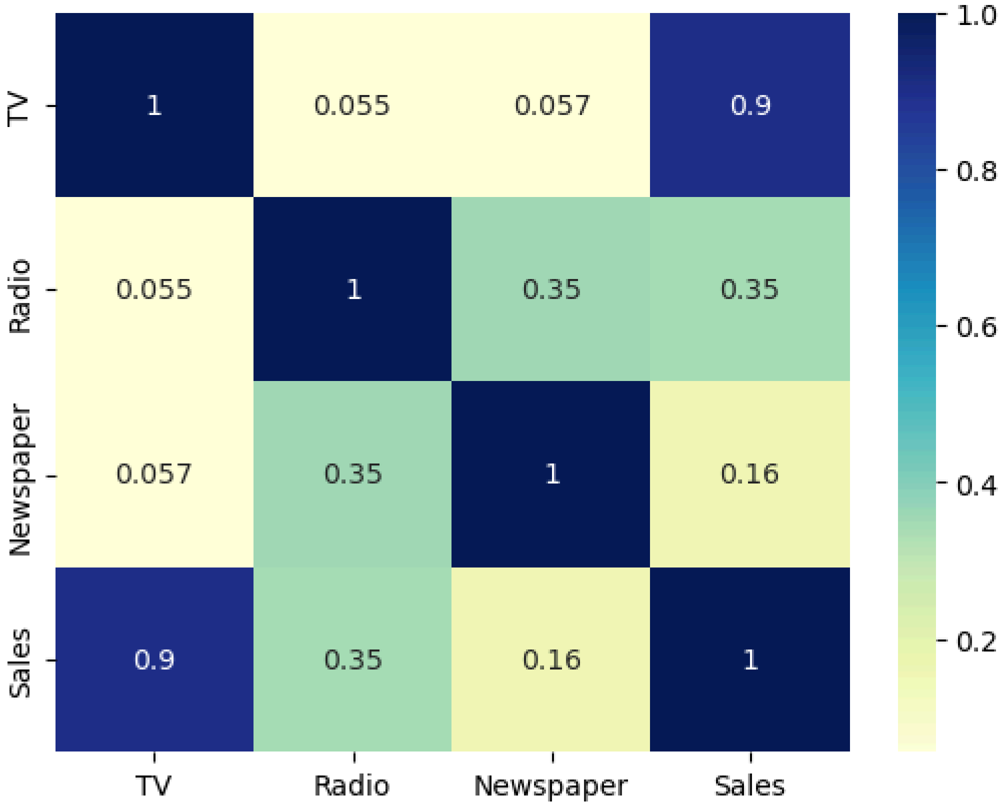


```
In [21]: snr.pairplot(data, x_vars=['TV', 'Newspaper', 'Radio'], y_vars='Sales', height=4, a
         plt.show()
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning:
The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



```
In [22]: sns.heatmap(data.corr(), cmap="YlGnBu", annot = True)
         plt.show()
```



In [23]:
```
X = data['TV']
Y = data['Sales']
```

In [25]:
```
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_si
```

In [26]:
```
X_train.head()
```

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

In [27]:
```
Y_train.head()
```

Out[27]:
```
74     17.0
3      16.5
185    22.6
26     15.0
90     14.0
Name: Sales, dtype: float64
```

In [28]:
```
import statsmodels.api as sm
```

In [30]:
```
X_train_sm = sm.add_constant(X_train)
lr = sm.OLS(Y_train, X_train_sm).fit()
```

In [31]:
```
lr.params
```

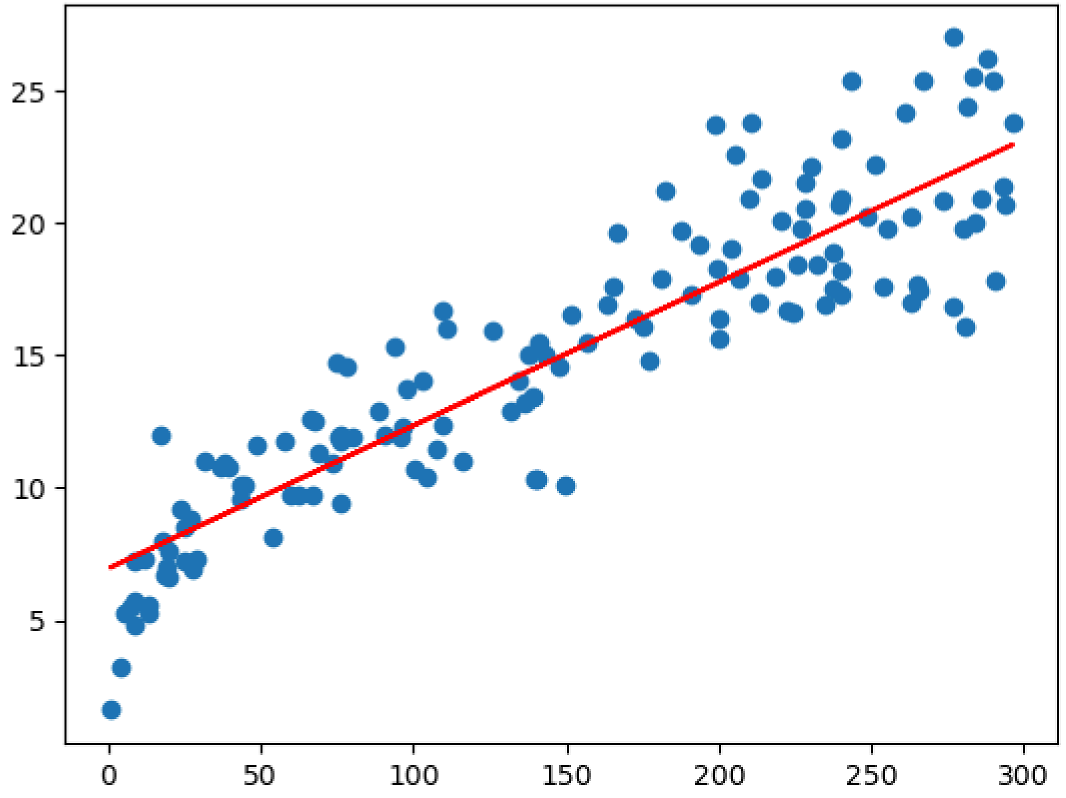
Out[31]:
```
const    6.948683
TV       0.054546
dtype: float64
```

In [32]:
```
print(lr.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.816
Model:                            OLS   Adj. R-squared:                  0.814
Method:                 Least Squares   F-statistic:                     611.2
Date:                Fri, 20 Dec 2024   Prob (F-statistic):           1.52e-52
Time:                        13:08:46   Log-Likelihood:                -321.12
No. Observations:                 140   AIC:                             646.2
Df Residuals:                     138   BIC:                             652.1
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          6.9487      0.385     18.068      0.000       6.188       7.709
TV             0.0545      0.002     24.722      0.000       0.050       0.059
==============================================================================
Omnibus:                        0.027   Durbin-Watson:                   2.196
Prob(Omnibus):                  0.987   Jarque-Bera (JB):                0.150
Skew:                          -0.006   Prob(JB):                        0.928
Kurtosis:                       2.840   Cond. No.                         328.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly s
pecified.
```

In [33]:

```
plt.scatter(X_train, Y_train)
plt.plot(X_train, 6.948 + 0.054*X_train, 'r')
plt.show()
```



In [34]:

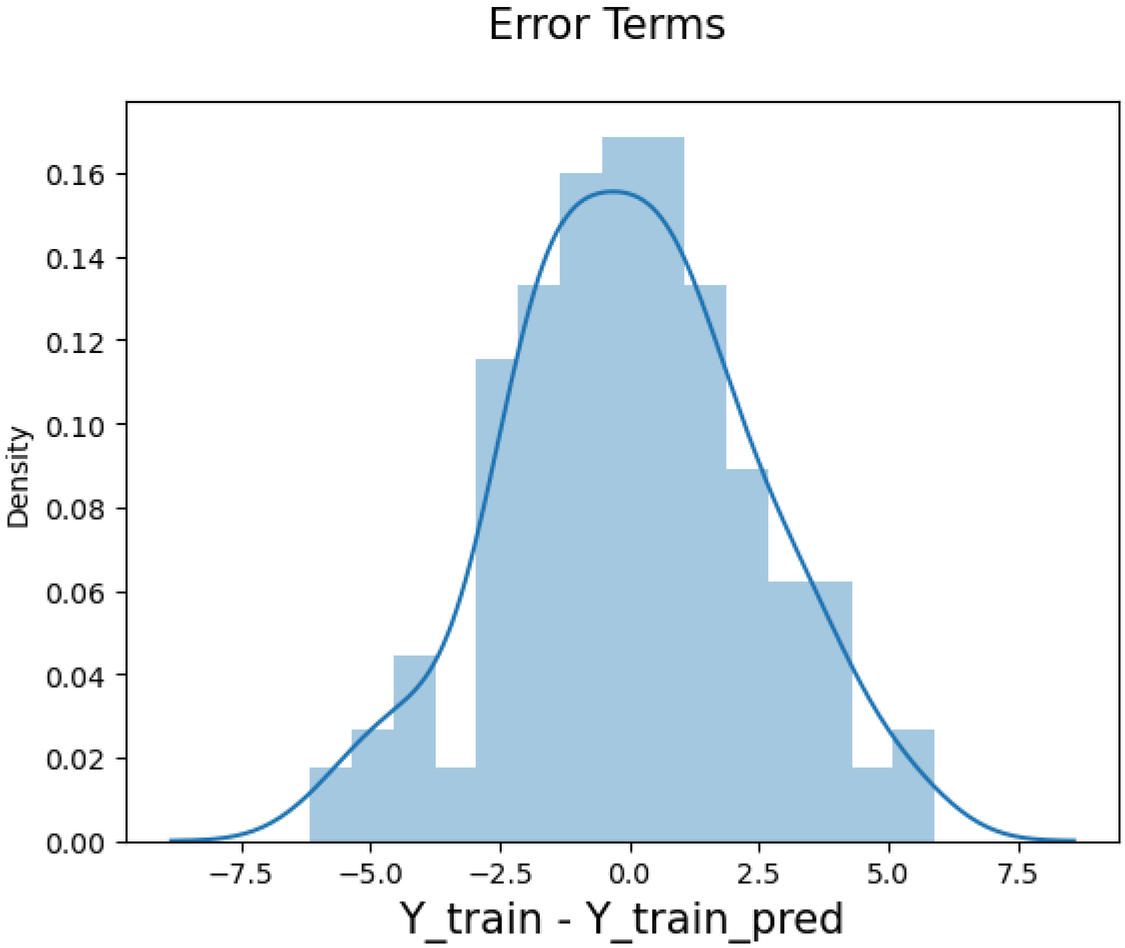
```
Y_train_pred = lr.predict(X_train_sm)
res =  (Y_train - Y_train_pred)
```

In [35]:

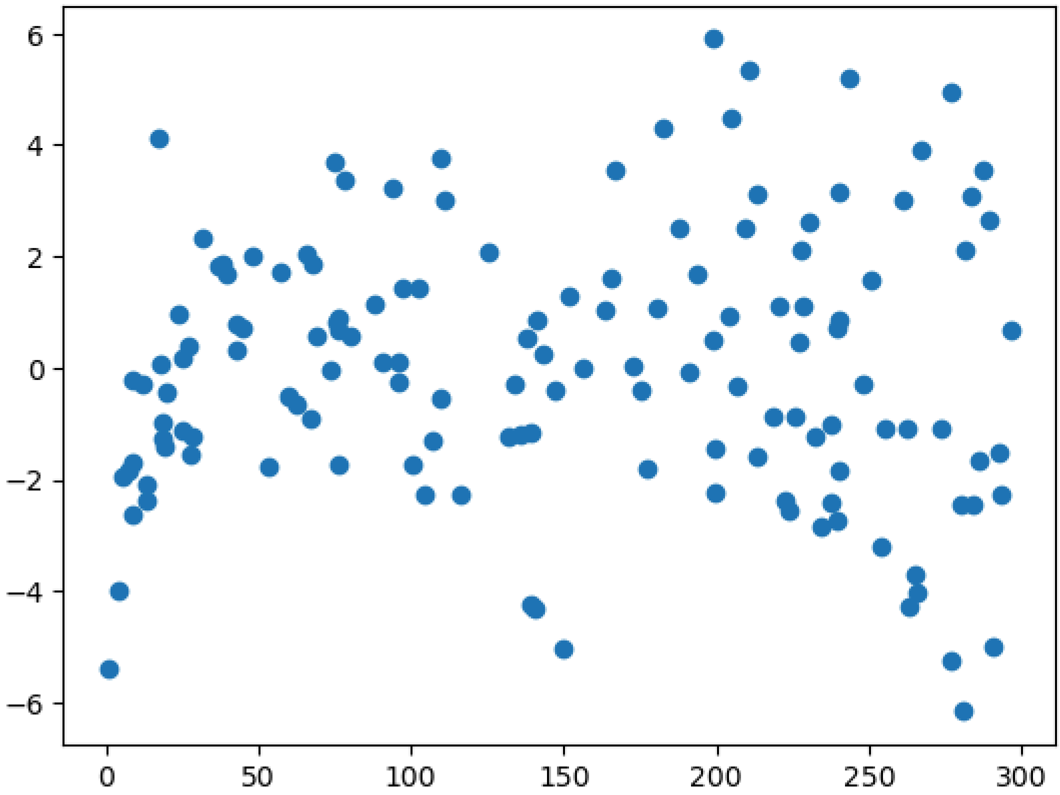
```
fig = plt.figure()
snr.distplot(res, bins = 15)
fig.suptitle('Error Terms', fontsize = 15)
```

```
plt.xlabel('Y_train - Y_train_pred', fontsize = 15)
plt.show()
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_6508\3636744094.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  snr.distplot(res, bins = 15)
```



```
In [36]: plt.scatter(X_train, res)
         plt.show()
```



```
In [37]: X_test_sm = sm.add_constant(X_test)
         Y_pred = lr.predict(X_test_sm)
```

```
In [38]: Y_pred.head()
```

```
Out[38]: 126     7.374140
         104    19.941482
         99     14.323269
         92     18.823294
         111    20.132392
         dtype: float64
```

```
In [40]: from sklearn.metrics import mean_squared_error
         from sklearn.metrics import r2_score
```

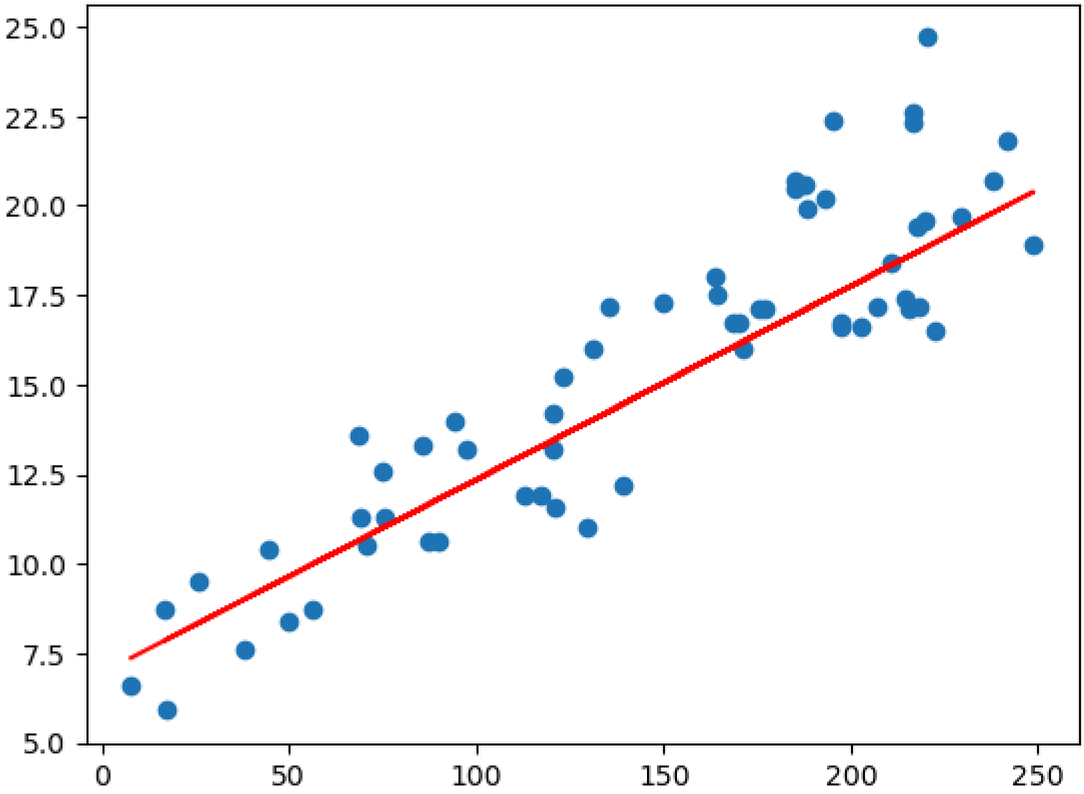
```
In [41]: np.sqrt(mean_squared_error(Y_test, Y_pred))
```

```
Out[41]: 2.019296008966232
```

```
In [42]: r_squared = r2_score(Y_test, Y_pred)
         r_squared
```

```
Out[42]: 0.7921031601245659
```

```
In [43]: plt.scatter(X_test, Y_test)
         plt.plot(X_test, 6.948 + 0.054 * X_test, 'r')
         plt.show()
```



```
In [ ]:
```