```
# prompt: import warnings

import warnings
warnings.filterwarnings('ignore')



import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns


advertising = pd.DataFrame(pd.read_csv("/content/advertising.csv"))


# prompt: advertising.head()

advertising.head()
```

	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

```
# prompt: advertising.shape()

advertising.shape
```

```
(200, 4)
```

```
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
```

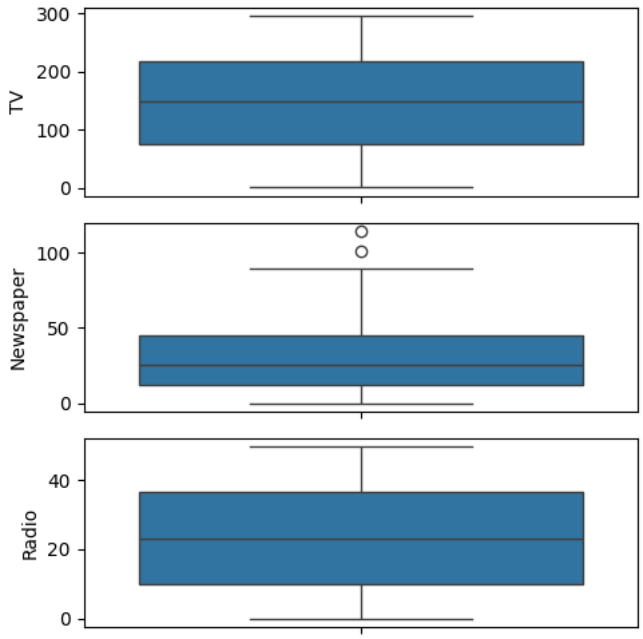
```
advertising.describe()
```

	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

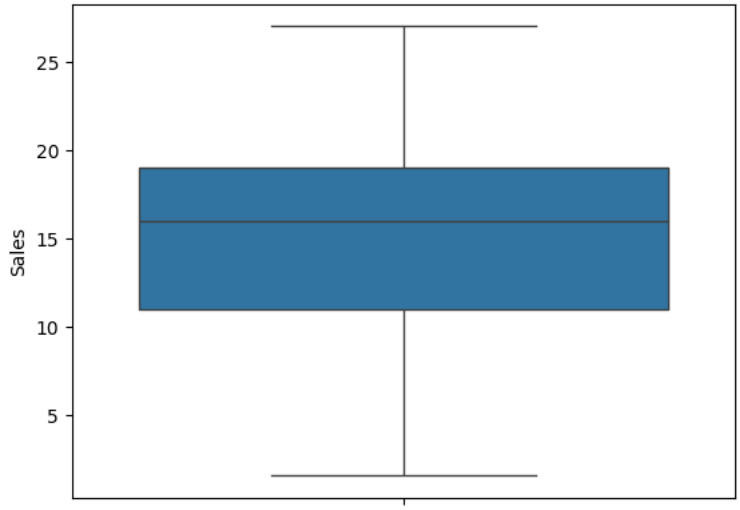
```
advertising.isnull().sum()*100/advertising.shape[0]
```

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

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




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




```
x = advertising['TV']
```

```
y = advertising['Sales']
```

```
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)
```

```
x_train.head()
```

```
24      62.3
176    248.4
40     202.5
52     216.4
94     107.4
Name: TV, dtype: float64
```

```
y_train.head()
```

```
24      9.7
176    20.2
40     16.6
52     22.6
94     11.5
Name: Sales, dtype: float64
```

```
import statsmodels.api as sm
```

```
x_train_sm = sm.add_constant(x_train)
```

```
lr = sm.OLS(y_train, x_train_sm).fit()
```

```
lr.params
```

```
const    6.960347
TV       0.054110
dtype: float64
```

```
print(lr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.822
Model:                            OLS   Adj. R-squared:                  0.820
Method:                 Least Squares   F-statistic:                     635.3
Date:                Wed, 31 Jan 2024   Prob (F-statistic):           1.69e-53
Time:                        09:24:28   Log-Likelihood:                -304.39
No. Observations:                 140   AIC:                             612.8
Df Residuals:                     138   BIC:                             618.7
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          6.9603      0.362     19.235      0.000       6.245       7.676
TV             0.0541      0.002     25.205      0.000       0.050       0.058
==============================================================================
Omnibus:                        0.582   Durbin-Watson:                   2.123
Prob(Omnibus):                  0.748   Jarque-Bera (JB):                0.245
Skew:                           0.002   Prob(JB):                        0.885
Kurtosis:                       3.205   Cond. No.                         337.
==============================================================================

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

```
y_train_pred = lr.predict(x_train_sm)
res = (y_train - y_train_pred)
```

```
x_test_sm = sm.add_constant(x_test)
y_pred = lr.predict(x_test_sm)
```

```
y_pred.head()
```

```
126     7.382404
120    14.606074
83     10.661464
134     8.957002
37     11.002356
dtype: float64
```

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

```
np.sqrt(mean_squared_error(y_test, y_pred))
```

```
2.6498686943078926
```

```
r_squared = r2_score(y_test, y_pred)
r_squared
```

```
0.78588599097311
```