In [2]:
```python
#import
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

In [3]:
```python
#load the dataset
```

In [4]:
```python
data_path = r"C:\Users\ARUTHRA D\Downloads\advertising.csv"
df = pd.read_csv(data_path)
```

In [5]:
```python
# Display the first few rows of the dataset
```

In [6]:
```python
print(df.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
```

In [7]:
```python
# data preprocessing
# Check for missing values in the dataset
```

In [8]:
```python
print(df.isnull().sum())
```

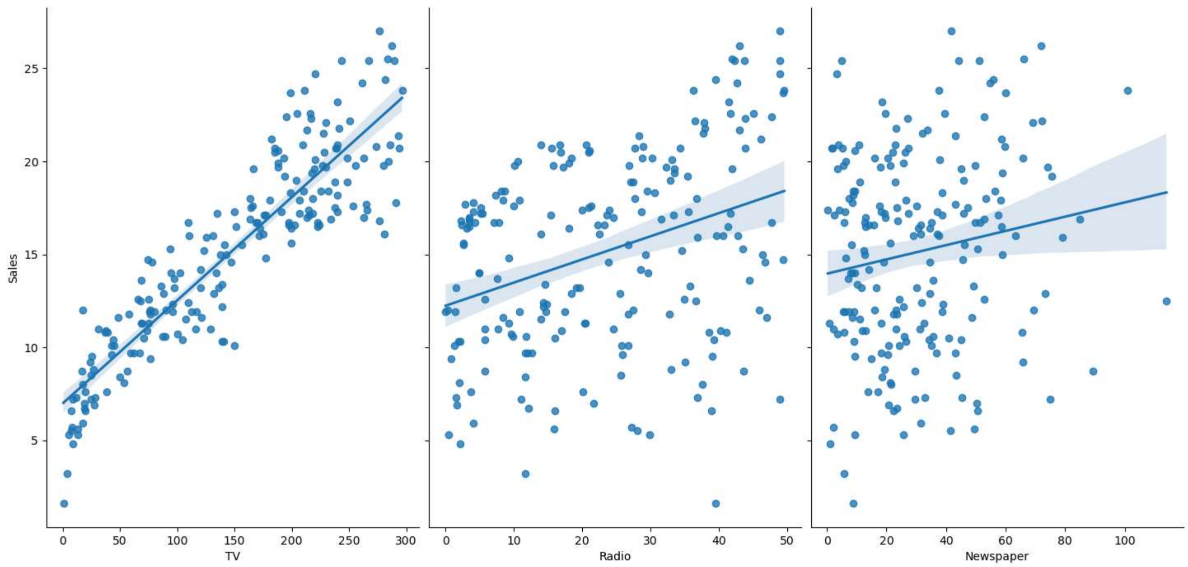
```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

In [9]:
```python
# Exploratory Data Analysis (EDA)
# Visualize the relationships between features and the target variable
```

In [10]:

```python
sns.pairplot(df, x_vars=['TV', 'Radio', 'Newspaper'], y_vars='Sales', height=7
plt.show()
```

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

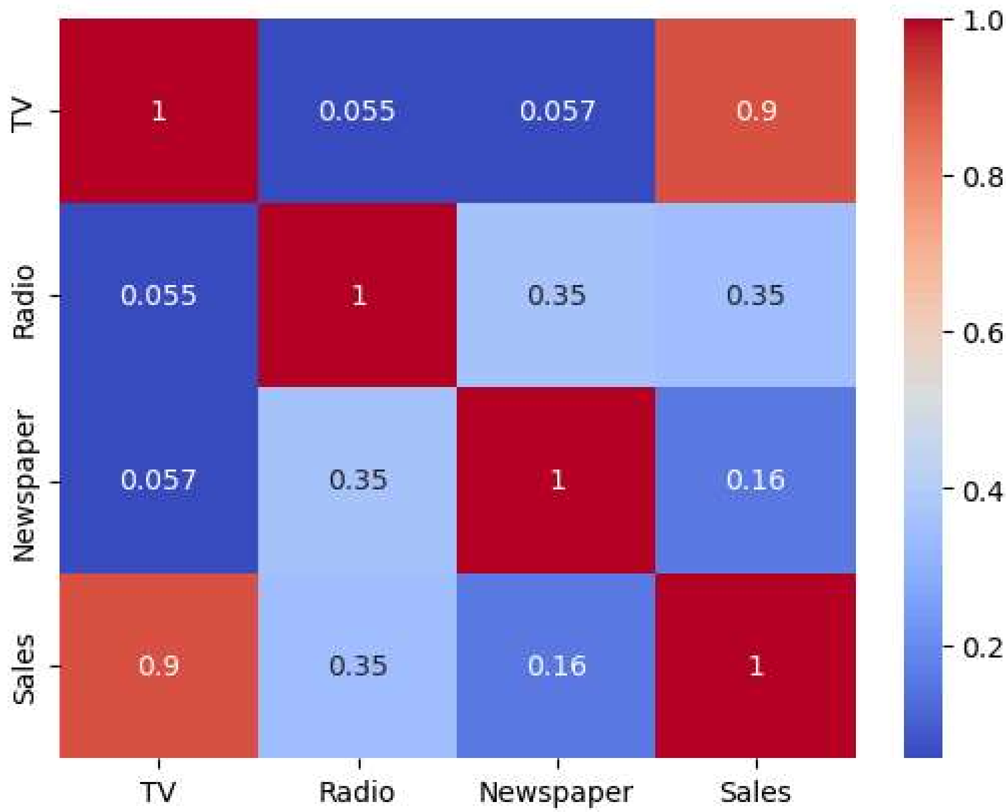


In [11]:

```python
# Generate a correlation matrix to see the relationships between variables
```

In [12]:

```python
corr = df.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.show()
```



In [13]:
```
# Feature Selection
# Define the features (independent variables) and the target (dependent variab
```

In [14]:
```
X = df[['TV', 'Radio', 'Newspaper']]
y = df['Sales']
```

In [15]:
```
# Model Selection and Training
# Split the dataset into training and testing sets
```

In [16]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
```

In [17]:
```
# Initialize the Linear Regression model
```

In [18]:
```
model = LinearRegression()
```

In [19]:
```
# Train the model on the training data
```

In [20]:
```
model.fit(X_train, y_train)
```

Out[20]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [21]:
```
# Model Evaluation
# Make predictions on the test data
```

In [22]:
```
y_pred = model.predict(X_test)
```

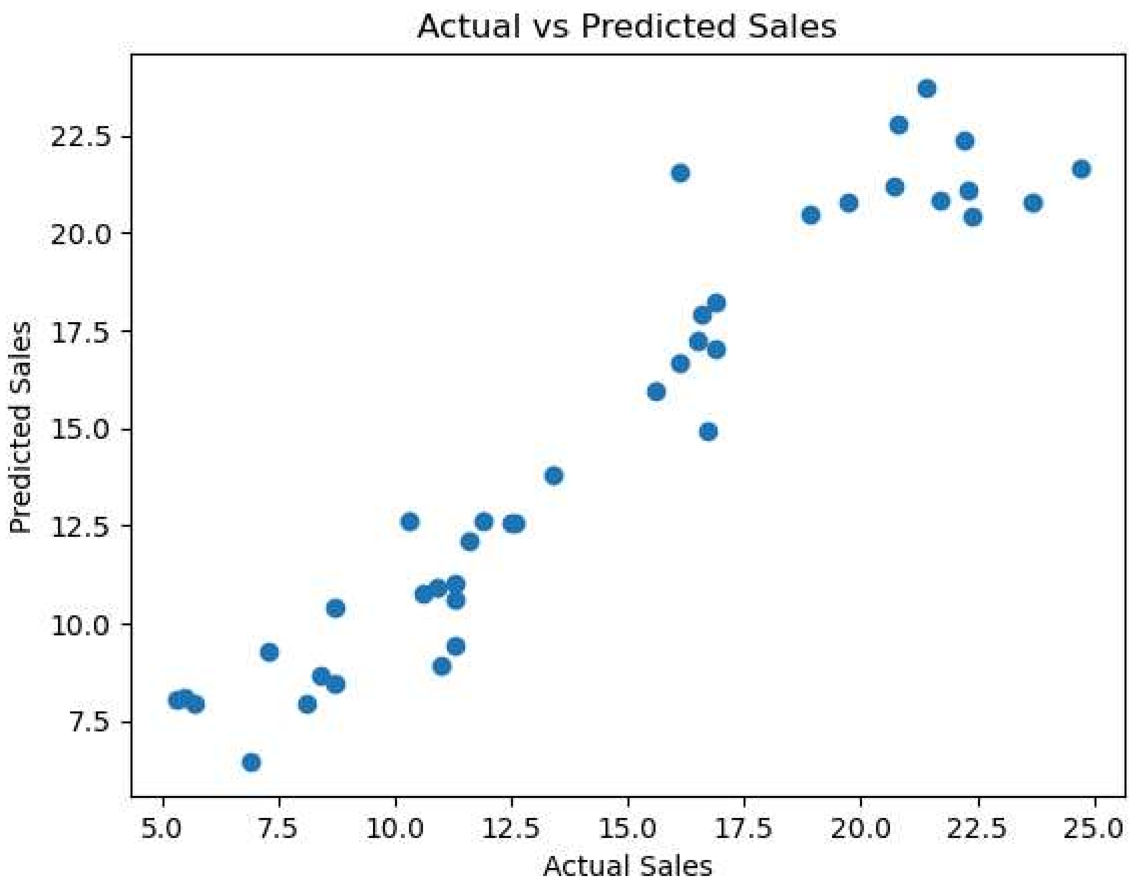
In [23]:
```
# Calculate evaluation metrics
```

In [24]:
```
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"Root Mean Squared Error: {rmse}")
print(f"R^2 Score: {r2}")
```

```
Mean Squared Error: 2.9077569102710905
Root Mean Squared Error: 1.7052146229349228
R^2 Score: 0.9059011844150826
```

In [25]:
```
# Plotting the actual vs predicted values
```

In [26]:
```
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Sales")
plt.ylabel("Predicted Sales")
plt.title("Actual vs Predicted Sales")
plt.show()
```



In [27]:
```
# Prediction and Optimization
# Example: Predicting sales for a new set of advertising expenditures
```

In [28]:
```
new_data = pd.DataFrame({
    'TV': [230.1, 44.5],
    'Radio': [37.8, 39.3],
    'Newspaper': [69.2, 45.1]
})
```

In [29]:
```
# Make predictions for the new data
```

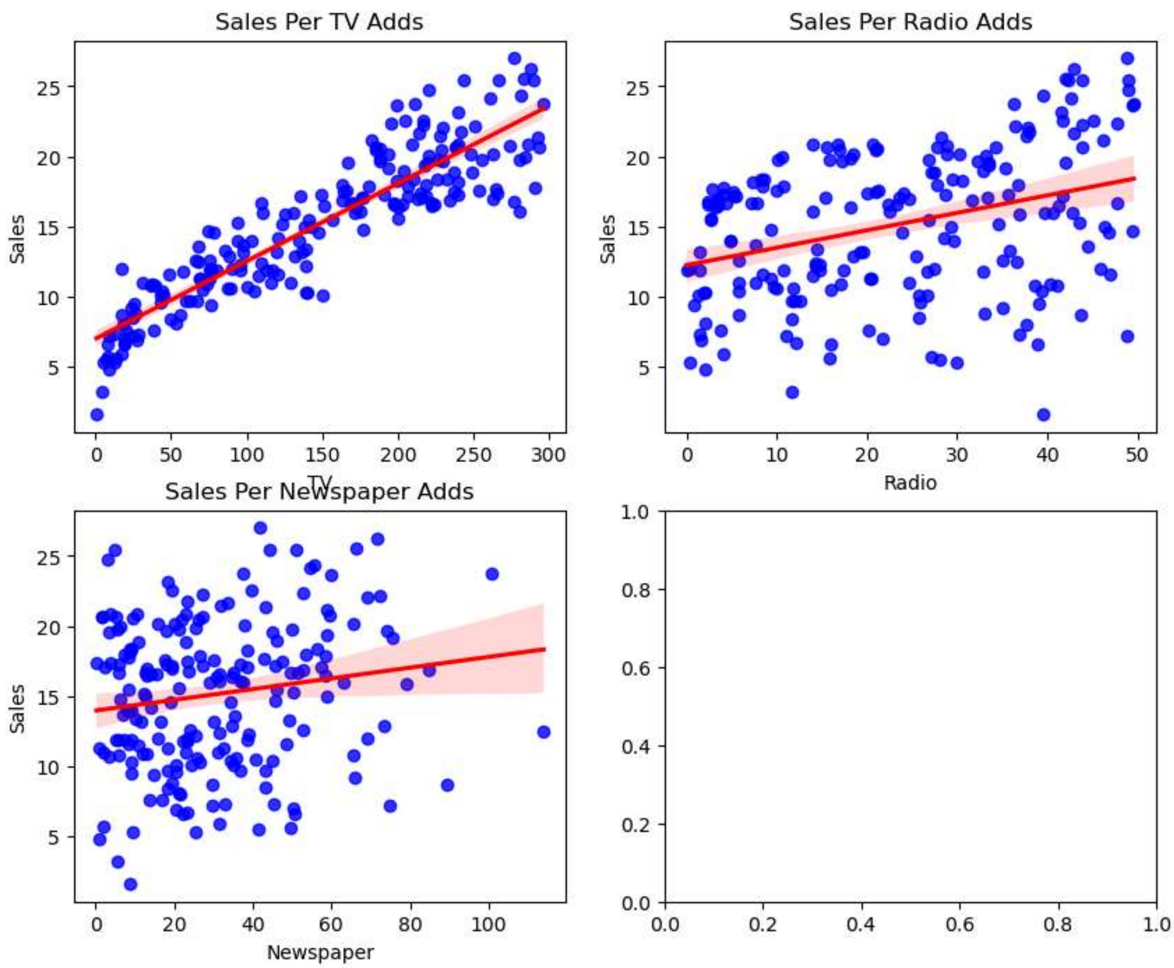
In [30]:
```python
print(f"Predicted Sales: {new_predictions}")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[30], line 1
----> 1 print(f"Predicted Sales: {new_predictions}")

NameError: name 'new_predictions' is not defined
```

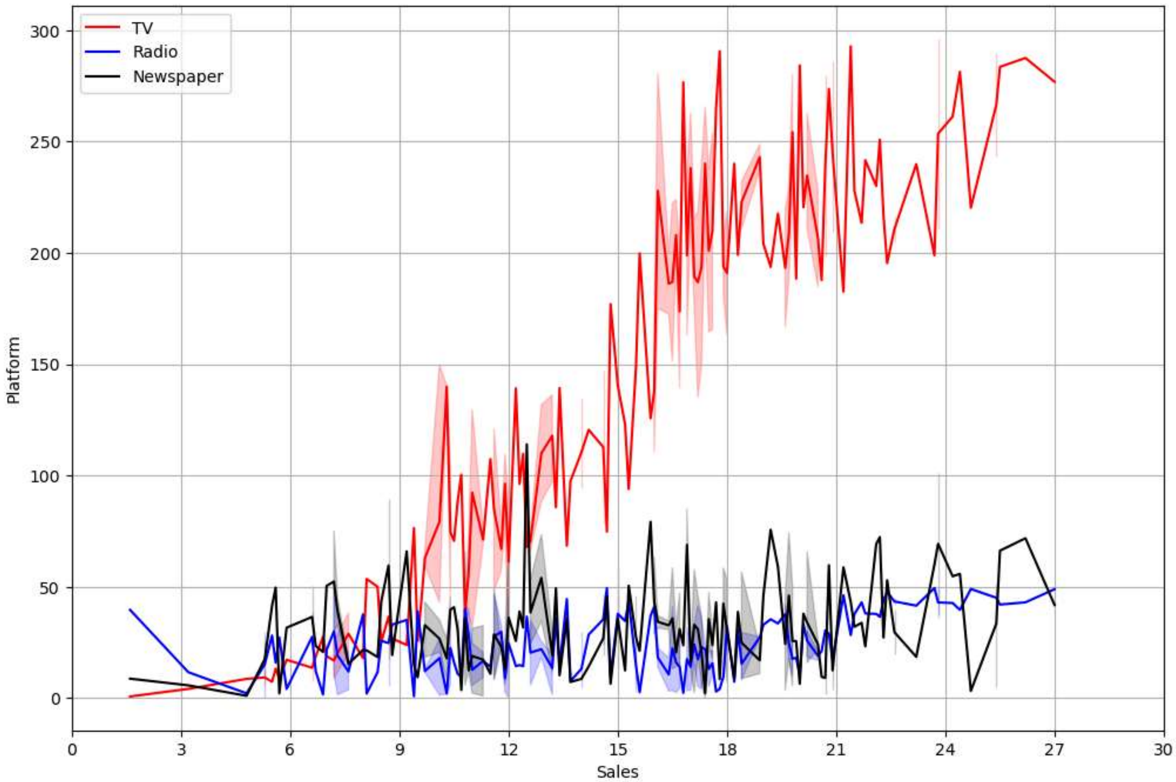
In [31]:
```python
new_predictions = model.predict(new_data)
print(f"Predicted Sales: {new_predictions}")
```

```
Predicted Sales: [21.37254028 11.30252447]
```

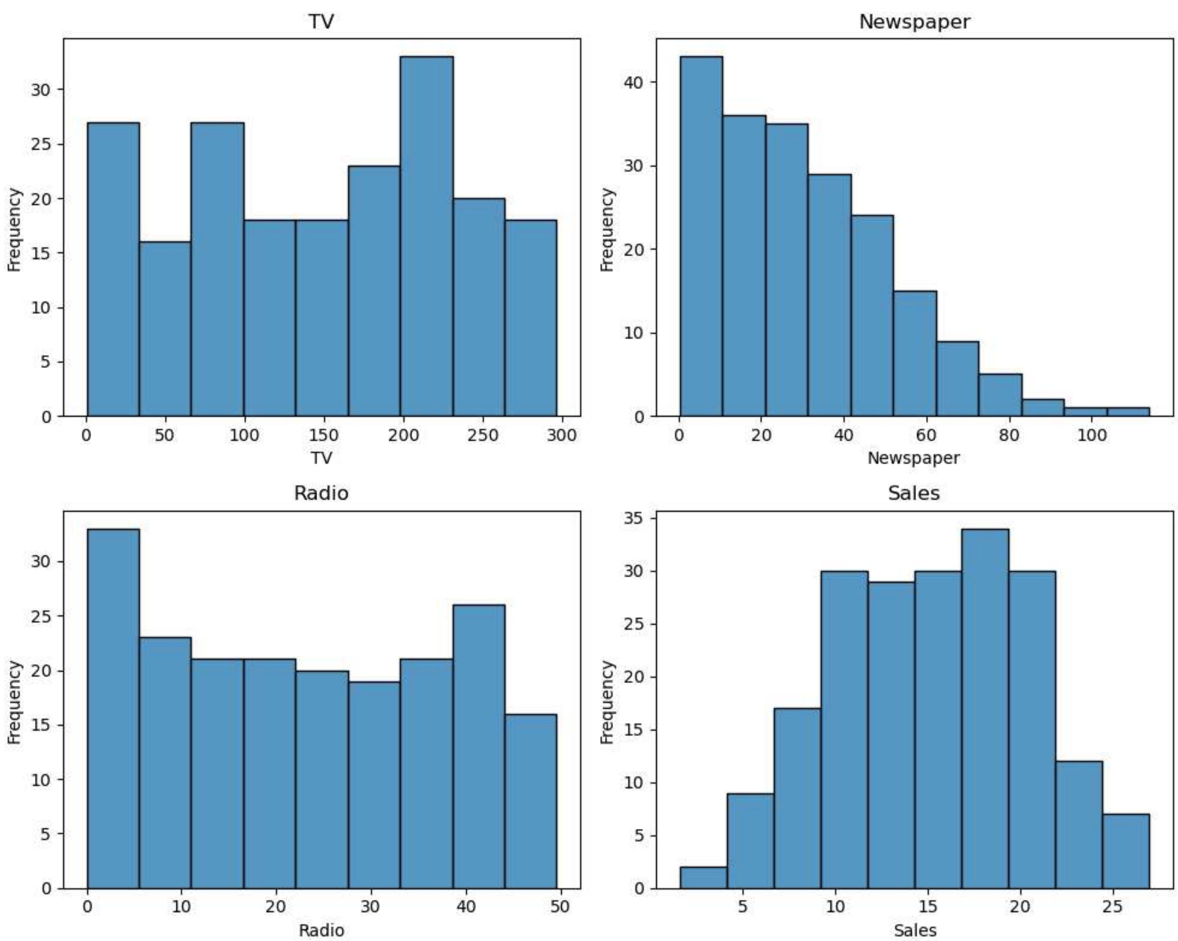
In [32]:
```
fig,ax=plt.subplots(nrows=2,ncols=2,figsize=(10,8))
#line_kws parameter is used to set the line color to red, while the scatter_kw
sns.regplot(x='TV',y='Sales',data=df,ax=ax[0][0],line_kws={'color': 'red'}, sc
ax[0][0].set_xlabel('TV')
ax[0][0].set_ylabel('Sales')
ax[0][0].set_title('Sales Per TV Adds')

sns.regplot(x='Radio',y='Sales',data=df,ax=ax[0][1],line_kws={'color': 'red'},
ax[0][1].set_xlabel('Radio')
ax[0][1].set_ylabel('Sales')
ax[0][1].set_title('Sales Per Radio Adds')

sns.regplot(x='Newspaper',y='Sales',data=df,ax=ax[1][0],line_kws={'color': 're
ax[1][0].set_xlabel('Newspaper')
ax[1][0].set_ylabel('Sales')
ax[1][0].set_title('Sales Per Newspaper Adds')

plt.show()
```



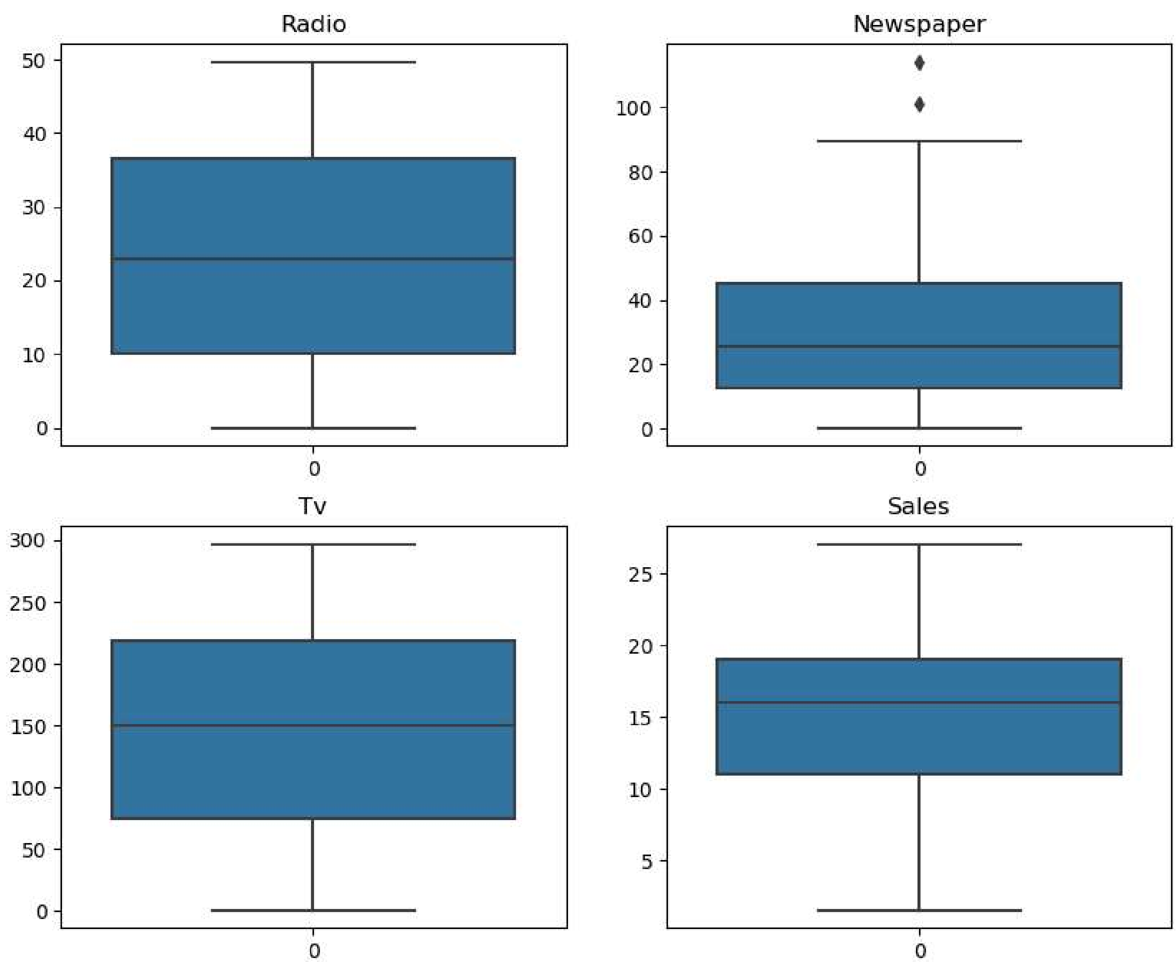
In [33]:
```
plt.figure(figsize=(12,8))

sns.lineplot(x='Sales',y='TV',data=df,color='r',label='TV')
sns.lineplot(x='Sales',y='Radio',data=df,color='blue',label='Radio')
sns.lineplot(x='Sales',y='Newspaper',data=df,color='black',label='Newspaper')
plt.xlabel('Sales')
plt.xticks(np.arange(0,33,3))
plt.ylabel('Platform')

plt.grid()
plt.show()
```



In [34]:
```
fig,ax=plt.subplots(nrows=2,ncols=2,figsize=(10,8))

sns.histplot(df['TV'],ax=ax[0][0])
ax[0][0].set_xlabel('TV')
ax[0][0].set_ylabel('Frequency')
ax[0][0].set_title('TV')

sns.histplot(df['Newspaper'],ax=ax[0][1])
ax[0][1].set_xlabel('Newspaper')
ax[0][1].set_ylabel('Frequency')
ax[0][1].set_title('Newspaper')

sns.histplot(df['Radio'],ax=ax[1][0])
ax[1][0].set_xlabel('Radio')
ax[1][0].set_ylabel('Frequency')
ax[1][0].set_title('Radio')

sns.histplot(df['Sales'],ax=ax[1][1])
ax[1][1].set_xlabel('Sales')
ax[1][1].set_ylabel('Frequency')
ax[1][1].set_title('Sales')

plt.tight_layout()
plt.show()
```



In [35]:
```
fig,ax=plt.subplots(nrows=2,ncols=2,figsize=(10,8))

sns.boxplot(df['Radio'],ax=ax[0][0])
ax[0][0].set_title('Radio')

sns.boxplot(df['Newspaper'],ax=ax[0][1])
ax[0][1].set_title('Newspaper')

sns.boxplot(df['TV'],ax=ax[1][0])
ax[1][0].set_title('Tv')

sns.boxplot(df['Sales'],ax=ax[1][1])
ax[1][1].set_title('Sales')

plt.show()
```



In [36]:
```
def quant(df,col,dis):
    q1=df[col].quantile(0.25)
    q3=df[col].quantile(0.75)
    iqr=q3-q1

    low=q1-(iqr*dis)
    upp=q3+(iqr*dis)
    return low,upp
lower,upper=quant(df,'Newspaper',1.5)
out=(df['Newspaper']<lower) | (df['Newspaper']>upper)
df['Newspaper'][out].count()
```

Out[36]:
```
2
```

In [37]:
```
df=df[~out]
df.shape
```

Out[37]:
```
(198, 4)
```

In [40]:
```
x=df.drop('Sales',axis=1)
y=df['Sales']
trainx,testx,trainy,testy=train_test_split(x,y,test_size=0.3,random_state=42)
LR=LinearRegression()
LR.fit(trainx,trainy)
pre_test=LR.predict(testx)
pre_train=LR.predict(trainx)
print('Accuarcy on Testing',r2_score(testy,pre_test))
print('Accuarcy on Training',r2_score(trainy,pre_train))
```

```
Accuarcy on Testing 0.9151626818586047
Accuarcy on Training 0.8905033650164197
```

In [41]:
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.model_selection import RepeatedKFold
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.linear_model import  LinearRegression
from sklearn.linear_model import Ridge
from sklearn.metrics import r2_score
```

In [42]:
```
kf=F=KFold(n_splits=5,random_state=43,shuffle=True)
cv=cross_val_score(LR,x,y,cv=kf,n_jobs=-1)
print('Accuracy : ',cv.mean()*100)
```

```
Accuracy :  89.29787896667068
```

In [43]:
```
rkf=RepeatedKFold(n_splits=5,n_repeats=15,random_state=5)
cv1=cross_val_score(LR,x,y,cv=rkf,n_jobs=-1)
print('Accuracy : ',cv1.mean()*100)
```

```
Accuracy :  88.93013860665874
```

In [44]:
```
RL=Ridge()
RL.fit(trainx,trainy)
pre_test_r=RL.predict(testx)
pre_train_r=RL.predict(trainx)
print('Accuarcy on Testing',r2_score(testy,pre_test_r))
print('Accuarcy on Training',r2_score(trainy,pre_train_r))
```

```
Accuarcy on Testing 0.9151615807367005
Accuarcy on Training 0.8905033649212801
```

In [45]:
```
kf=F=KFold(n_splits=5,random_state=43,shuffle=True)
cv3=cross_val_score(RL,x,y,cv=kf,n_jobs=-1)
print('Accuracy : ',cv3.mean()*100)
```

```
Accuracy :  89.29788025233654
```

In [46]:
```
rkf=RepeatedKFold(n_splits=5,n_repeats=15,random_state=5)
cv4=cross_val_score(RL,x,y,cv=rkf,n_jobs=-1)
print('Accuracy : ',cv4.mean()*100)
```

```
Accuracy :  88.93014423341027
```

In [ ]:
```

```