Some Important Required Libraries

In [57]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

In [58]:
```
data=pd.read_csv("C:/Users/Dell/Downloads/Advertising.csv")
```

In [59]:
```
data
```

Out[59]:

	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 [60]:
```
data.shape
```

Out[60]:
```
(200, 5)
```

In [61]:
```
data.info
```

Out[61]:
```
<bound method DataFrame.info of      Unnamed: 0     TV  Radio  Newspaper  Sal
es
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 x 5 columns]>
```

In [62]:
```
data.head(10)
```

Out[62]:

	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
5	6	8.7	48.9	75.0	7.2
6	7	57.5	32.8	23.5	11.8
7	8	120.2	19.6	11.6	13.2
8	9	8.6	2.1	1.0	4.8
9	10	199.8	2.6	21.2	10.6

In [63]:
```
data.describe()
```

Out[63]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000	200.000000
mean	100.500000	147.042500	23.264000	30.554000	14.022500
std	57.879185	85.854236	14.846809	21.778621	5.217457
min	1.000000	0.700000	0.000000	0.300000	1.600000
25%	50.750000	74.375000	9.975000	12.750000	10.375000
50%	100.500000	149.750000	22.900000	25.750000	12.900000
75%	150.250000	218.825000	36.525000	45.100000	17.400000
max	200.000000	296.400000	49.600000	114.000000	27.000000

In [64]:

```
data.tail()
```

Out[64]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
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

In [65]:

```
  data.columns
```

Out[65]:

```
Index(['Unnamed: 0', 'TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')
```

In [66]:

```
data.dtypes
```

Out[66]:

```
Unnamed: 0      int64
TV            float64
Radio         float64
Newspaper     float64
Sales         float64
dtype: object
```

In [67]:

```
#Checking the null values in column
data.isnull().sum()
```

Out[67]:

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

In [68]:

```
data.count()
```

Out[68]:

```
Unnamed: 0    200
TV            200
Radio         200
Newspaper     200
Sales         200
dtype: int64
```

In [69]:
```
corr_matrix = data.corr()
corr_matrix
```

Out[69]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
Unnamed: 0	1.000000	0.017715	-0.110680	-0.154944	-0.051616
TV	0.017715	1.000000	0.054809	0.056648	0.782224
Radio	-0.110680	0.054809	1.000000	0.354104	0.576223
Newspaper	-0.154944	0.056648	0.354104	1.000000	0.228299
Sales	-0.051616	0.782224	0.576223	0.228299	1.000000

In [70]:
```
data['Sales'].value_counts()
```

Out[70]:
```
9.7     5
11.7    4
12.9    4
15.9    4
20.7    3
       ..
17.0    1
18.3    1
22.3    1
14.0    1
25.5    1
Name: Sales, Length: 121, dtype: int64
```

Data Splitting

In [71]:
```
# Split the data into input(x) and (y)variable
x = data[['TV','Radio','Newspaper']]
y = data['Sales']
```

In [72]:
```
# Splitting the dataset into train and test sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, rando
```

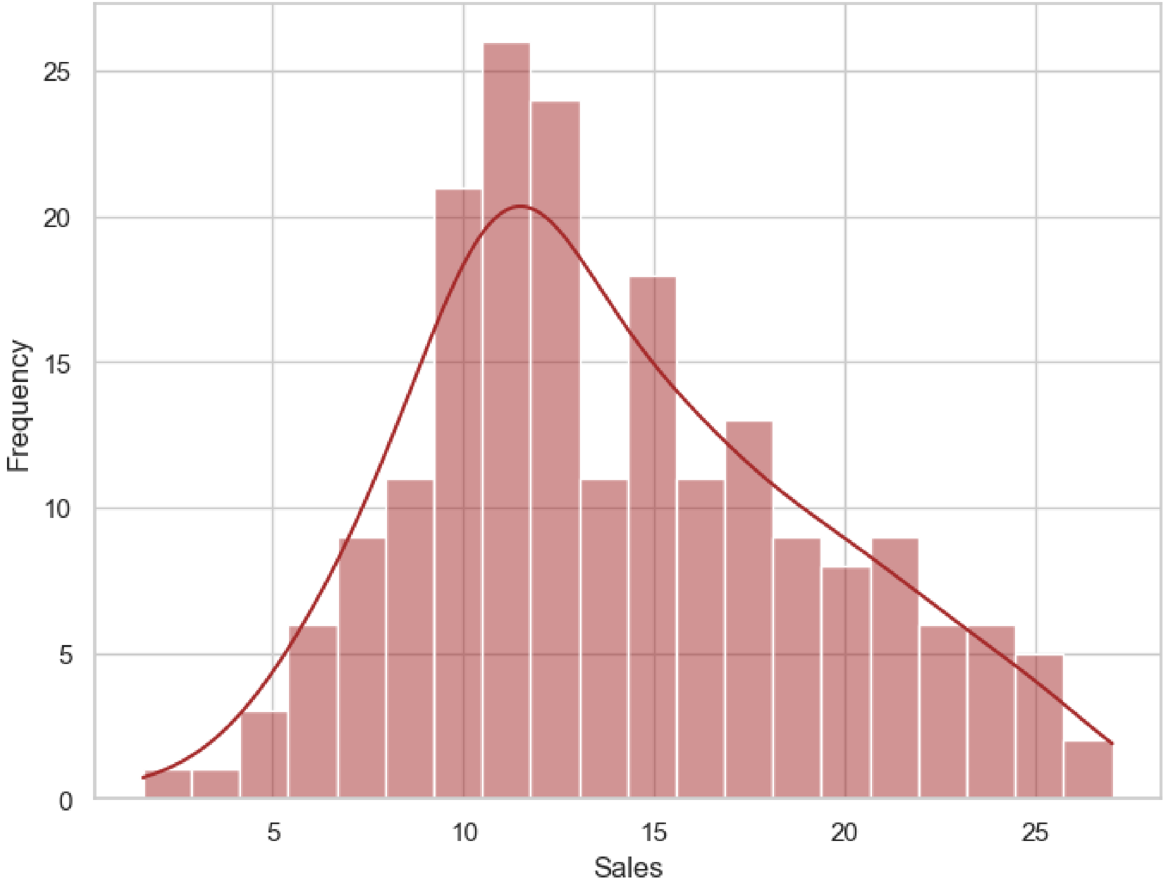
In [73]:
```
print(x.shape, x_train.shape, x_test.shape)
```

```
(200, 3) (160, 3) (40, 3)
```

Visualization Data

In [112]:

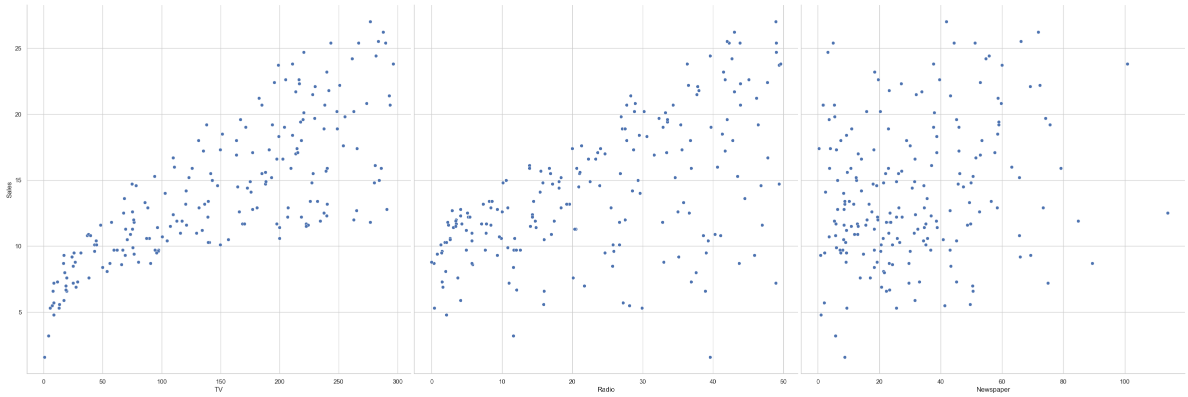
```
sns.set(style='whitegrid')
plt.figure(figsize=(8,6))
sns.histplot(data['Sales'], bins=20, color='brown', kde=True )
plt.xlabel('Sales')
plt.ylabel('Frequency')
plt.show()
```



In [75]:

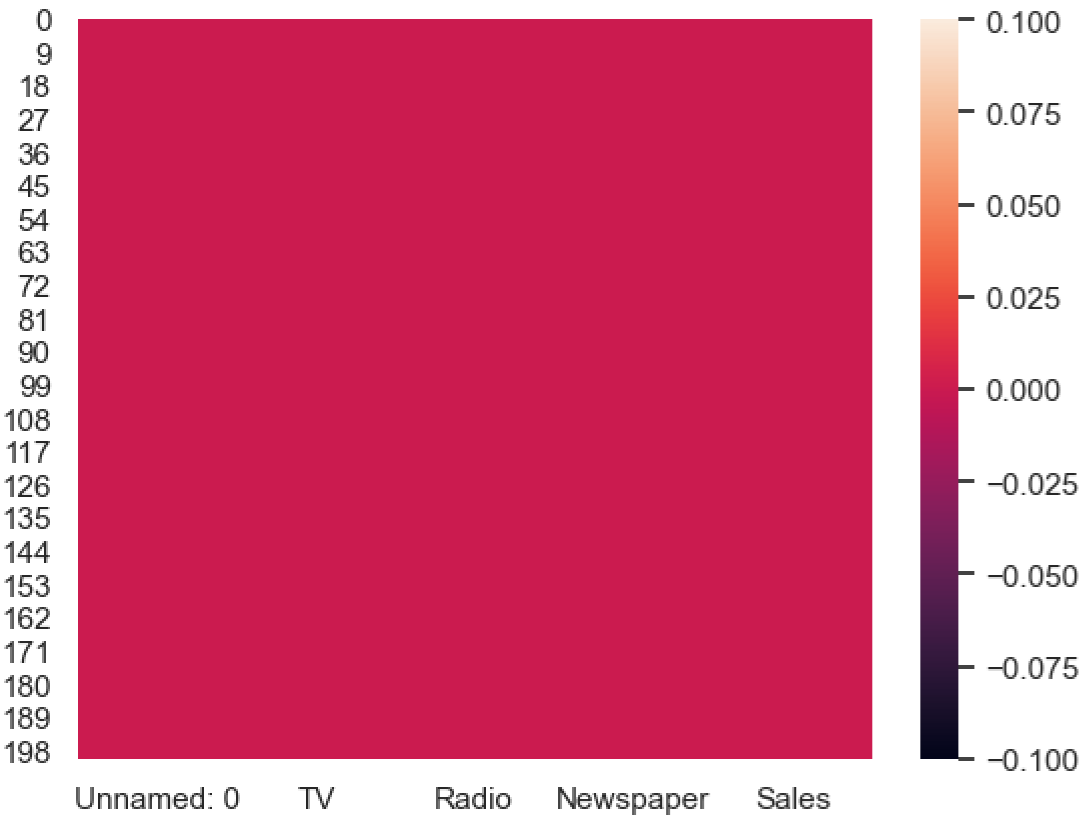
```
plt.figure(figsize=(6,6))
sns.pairplot(data, x_vars=['TV','Radio','Newspaper'],y_vars='Sales',height=10,
plt.show()
```

```
<Figure size 600x600 with 0 Axes>
```

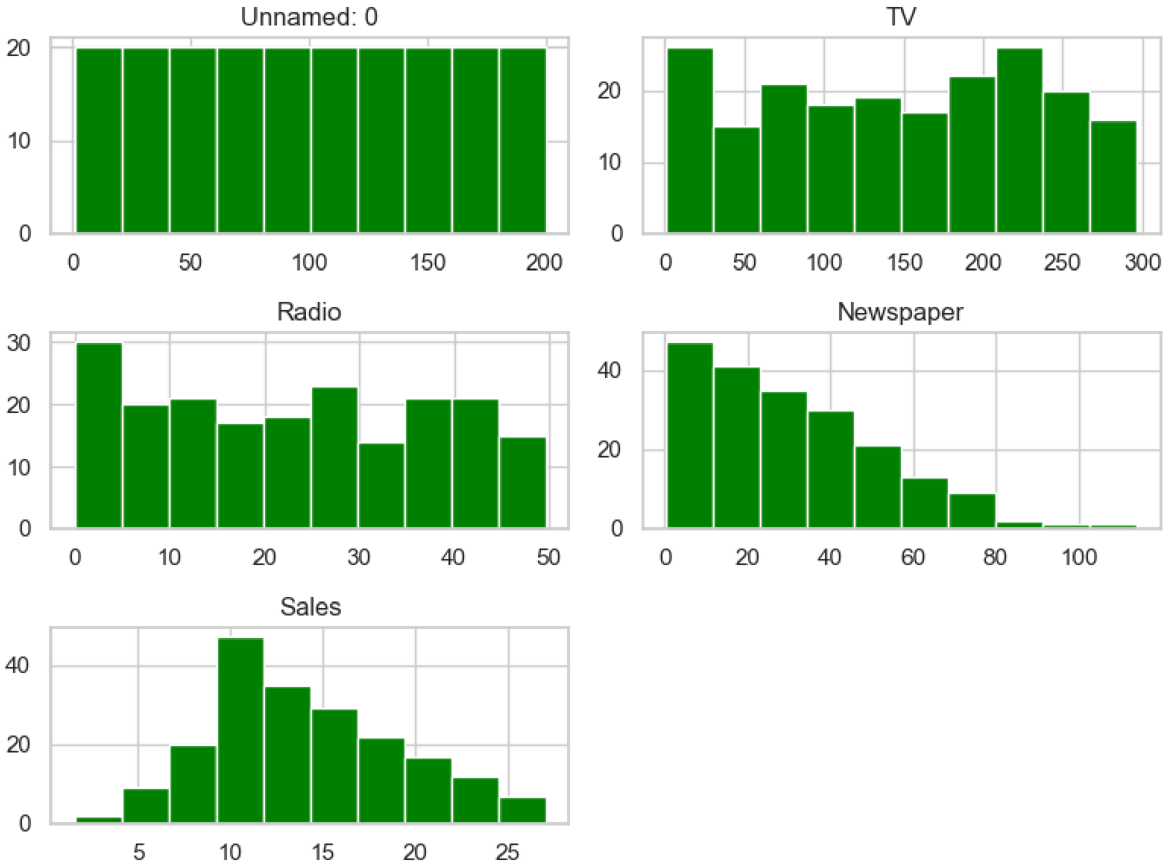


In [76]:
```
sns.heatmap(df.isnull())
```

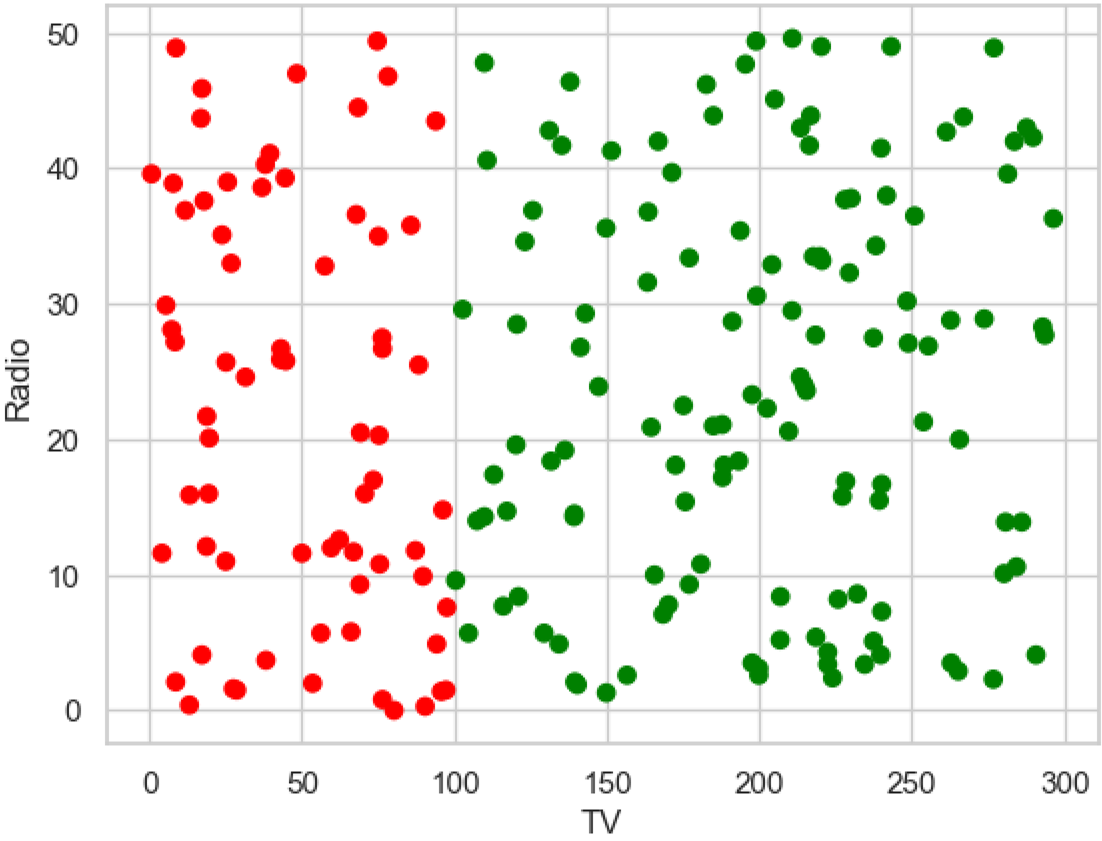
Out[76]:
```
<AxesSubplot: >
```



In [110]:
```
data.hist(bins=10,color='green', figsize=(8,6))
plt.tight_layout()
plt.show()
```

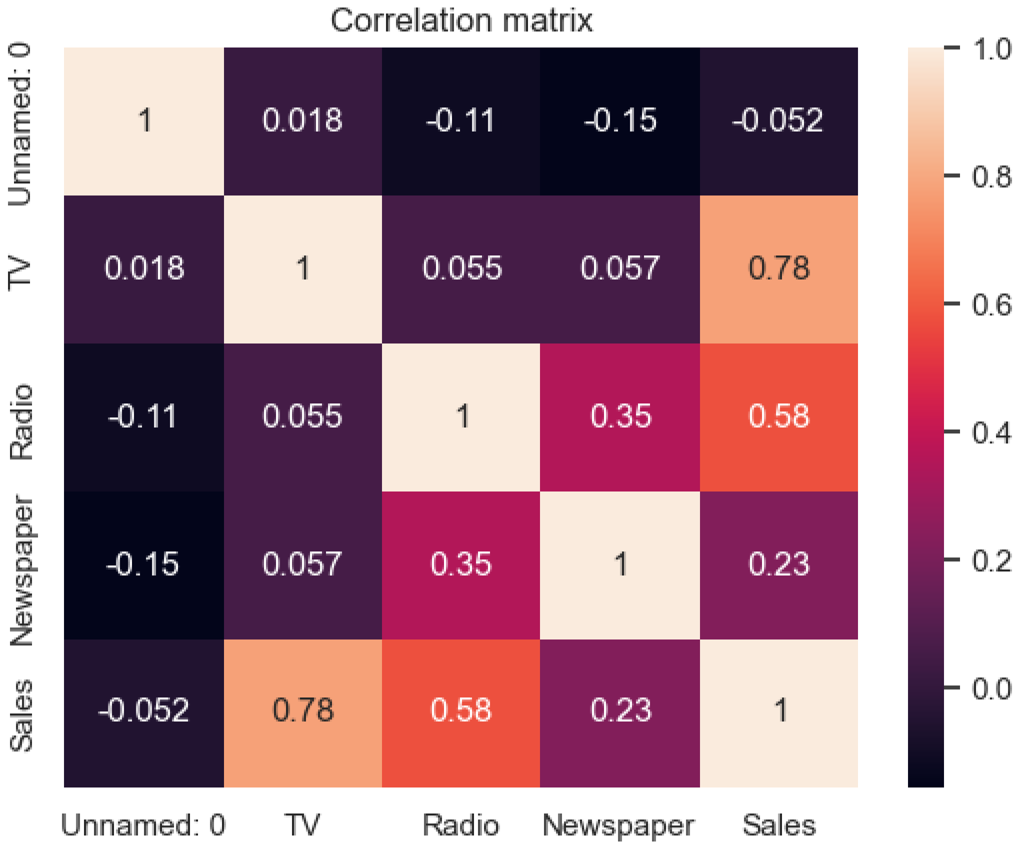


```
In [111]: plt.scatter(data['TV'], data['Radio'], c=['green' if length>=100 else 'red' fo
          plt.xlabel("TV")
          plt.ylabel("Radio")
          plt.show()
```



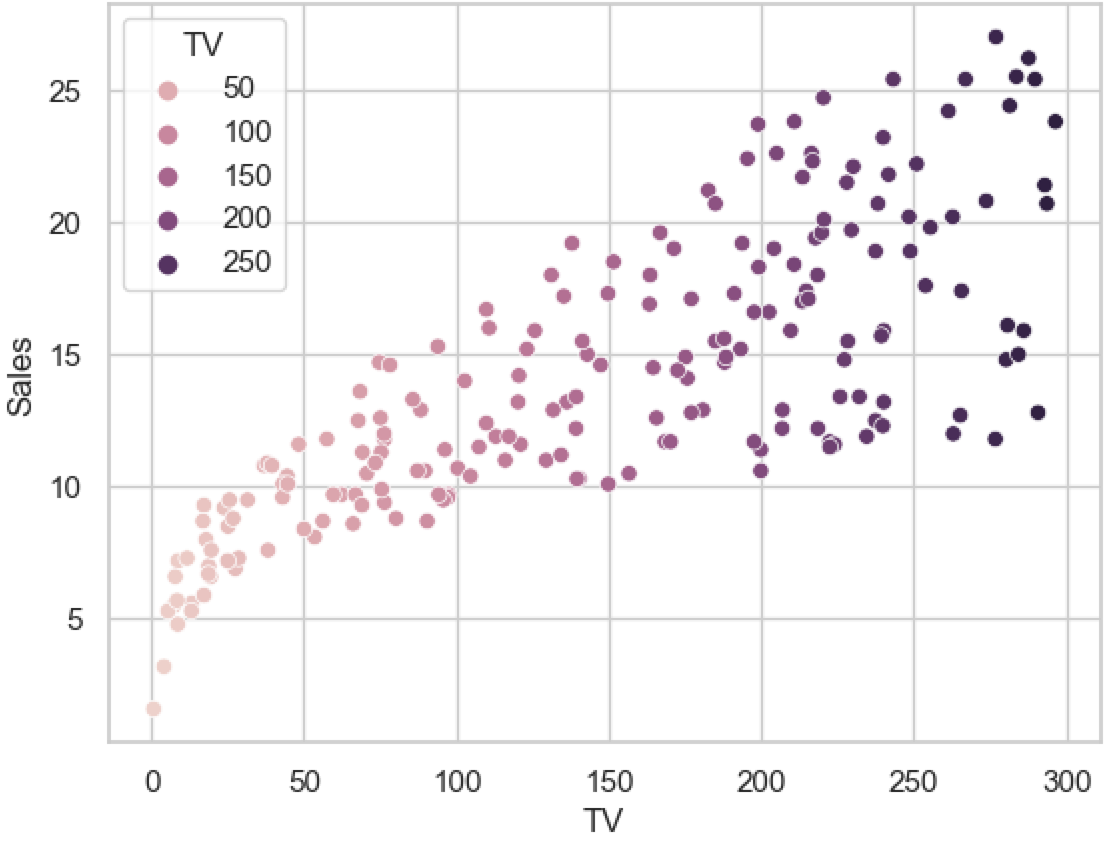
In [79]:

```
sns.heatmap(corr_matrix, annot = True)
plt.title("Correlation matrix")
plt.show()
```



```
In [80]: sns.scatterplot(x='TV', y='Sales', hue='TV', data=data)
```

```
Out[80]: <AxesSubplot: xlabel='TV', ylabel='Sales'>
```

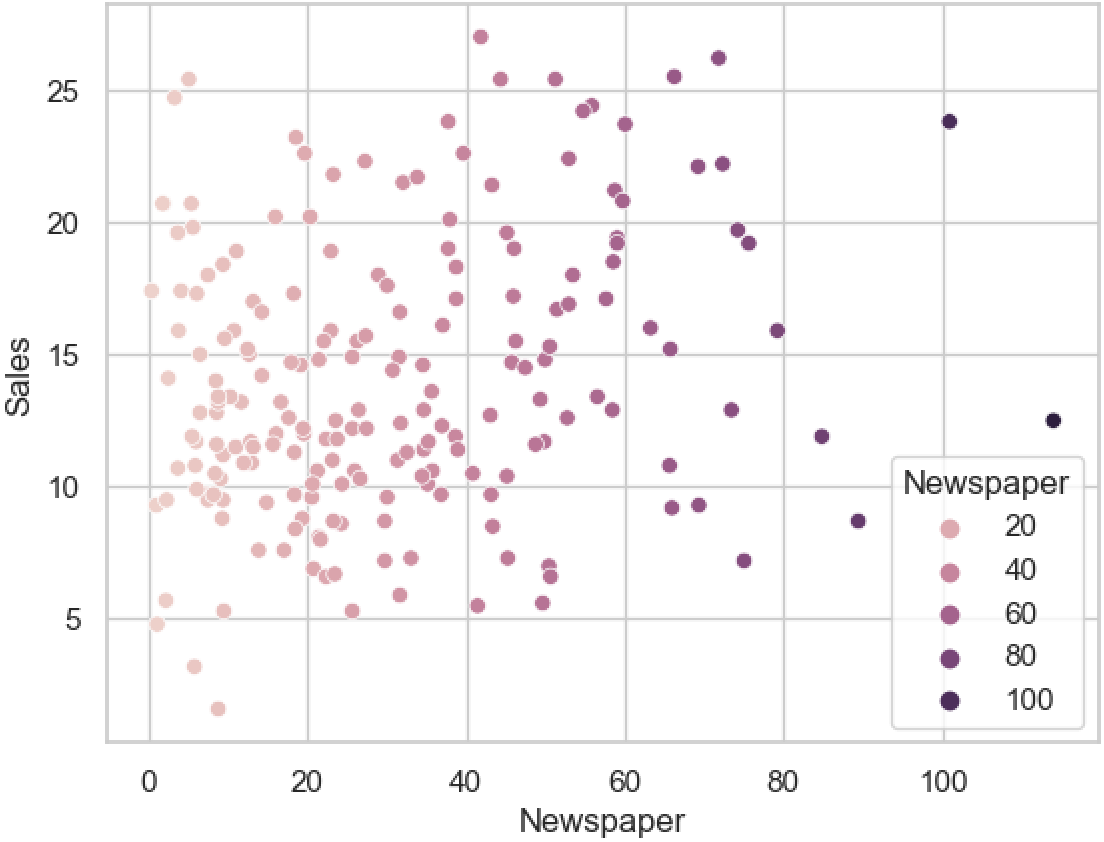


In [81]:

```
sns.scatterplot(x='Newspaper', y='Sales', hue ='Newspaper',data=data)
```

Out[81]:

```
<AxesSubplot: xlabel='Newspaper', ylabel='Sales'>
```



Training Model

In [82]:

```
# linear Regression
```

In [83]:

```
model = LinearRegression()
```

In [84]:

```
model.fit(x_train, y_train)
```

Out[84]:

```
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 [85]:

```
y_predict = model.predict(x_test)
```

In [86]:

```
# Evaluation
```

In [87]:

```
# Mean Squared Error
rmse = mean_squared_error(y_test, y_predict, squared = False)
rmse
```

Out[87]:

```
1.78159966153345
```

In [89]:

```
r2 = r2_score(y_test, y_predict)
r2
```

Out[89]:

```
0.899438024100912
```

In [93]:

```
# XGBRegression

from xgboost import XGBRegressor
regressor = XGBRegressor()
regressor.fit(x_train, y_train)
```

Out[93]:

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...)
```

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 [100]:

```
# Prediction of training data
tdata_predict = regressor.predict(x_train)
```

In [101]:

```
# R-Squared value
r2_test = r2_score(y_train, tdata_predict)
r2_test
```

Out[101]:

```
0.9999999400896089
```

In [ ]:

```
# Prediction of training data
darta_predict = regressor.predict(x_train)
```

In [102]:

```
# R-Squared value
r2_test = r2_score(y_train, rdata_predict)
r2_test
```

Out[102]:

```
0.9999999400896089
```

In [104]:

```
# Random Forest Regression
random = RandomForestRegressor(random_state=42)
random.fit(x_train, y_train)
```

Out[104]:

```
RandomForestRegressor(random_state=42)
```

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 [105]:

```
# Predicting on test set
y_predict = random.predict(x_test)
```

In [106]:

```
# Mean squared_error
rmse = mean_squared_error(y_test, y_predict, squared = False)
rmse
```

Out[106]:

```
0.7685910811348248
```

In [107]:

```
r2 = r2_score(y_test, y_predict)
r2
```

Out[107]:

```
0.9812843792541843
```