In [3]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import norm
```

In [4]:

```
dataset = pd.read_csv("Advertising.csv")
```

In [5]:

```
dataset.head()
```

Out[5]:

	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	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

In [6]:

```
dataset.describe()
```

Out[6]:

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

In [7]:

```
dataset.corr()
```

Out[7]:

	TV	Radio	Newspaper	Sales
TV	1.000000	0.054809	0.056648	0.782224
Radio	0.054809	1.000000	0.354104	0.576223
Newspaper	0.056648	0.354104	1.000000	0.228299
Sales	0.782224	0.576223	0.228299	1.000000

In [8]:

```
dataset.shape
```

Out[8]:

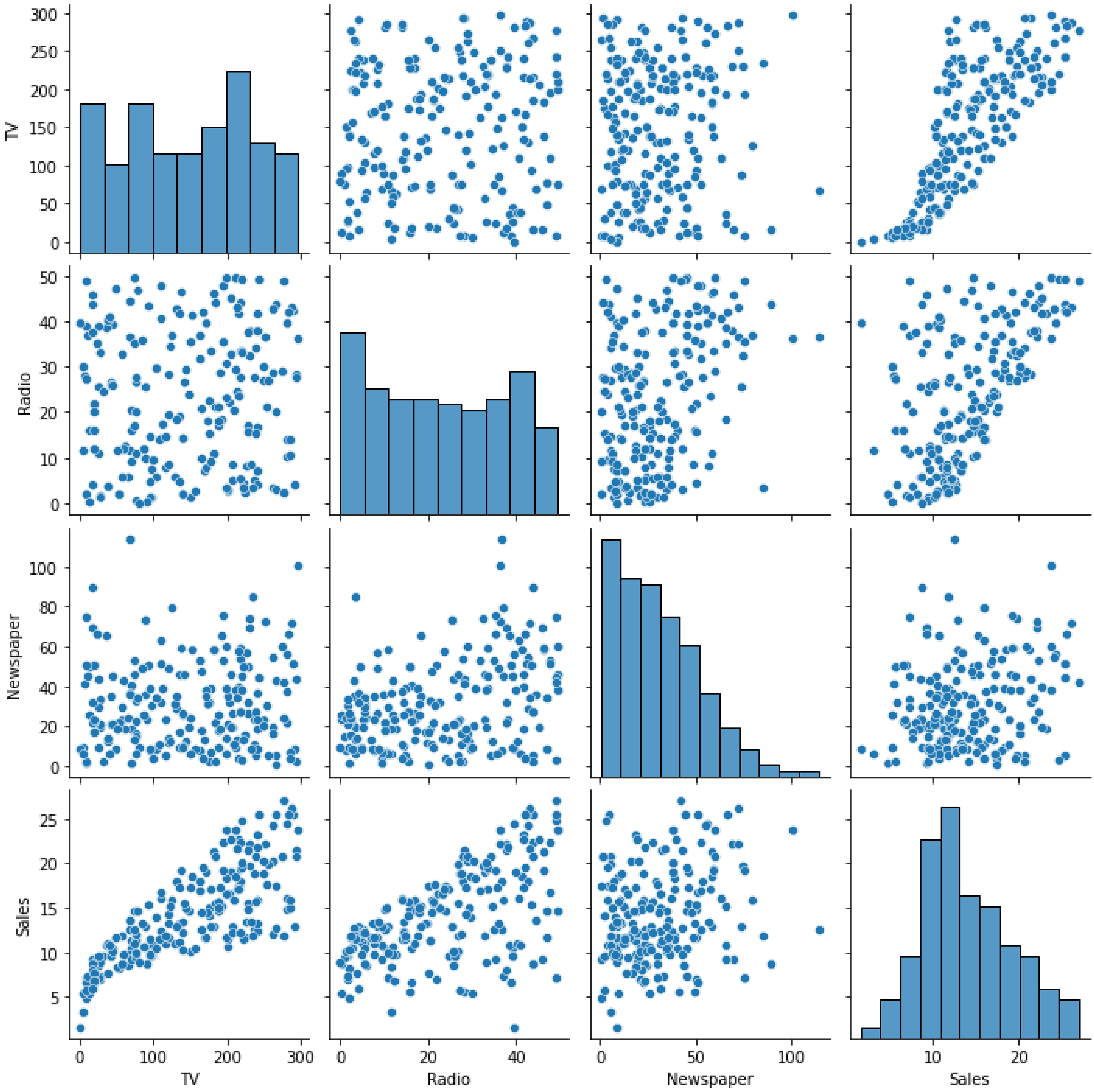
```
(200, 4)
```

In [9]:

```
sns.pairplot(dataset)
```

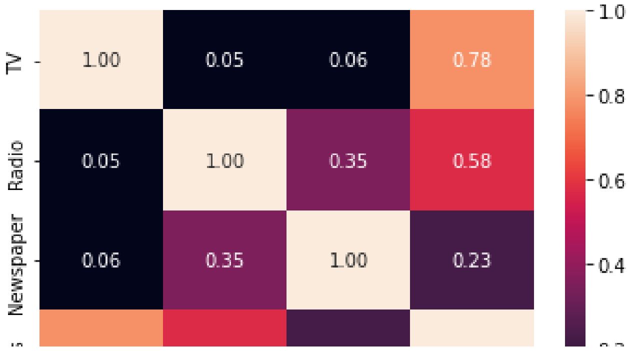
Out[9]:

```
<seaborn.axisgrid.PairGrid at 0x7f96d7f6c050>
```



In [10]:

```
sns.heatmap(dataset.corr(),annot=True,fmt=".2f")
plt.show()
```



In [11]:

```
x = dataset[['TV', 'Radio', 'Newspaper']]
y = dataset['Sales']
```

In [12]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=13
)
```

In [13]:

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression(fit_intercept=True)
regressor.fit(x_train, y_train)
```

Out[13]:

```
LinearRegression()
```

In [14]:

```
coef_df = pd.DataFrame(regressor.coef_, x.columns, columns=['Coefficient'])
coef_df
```

Out[14]:

	Coefficient
TV	0.045958
Radio	0.185460
Newspaper	-0.002729

In [15]:

```
regressor.intercept_
```

Out[15]:

```
2.996463132658773
```

In [16]:

```
y_pred = regressor.predict(x_train)
y_pred
```

Out[16]:

```
array([ 9.1217473 , 18.97702787, 21.02762902,  3.60283185, 15.60500758,
       21.08868438, 19.05341938, 10.05627154, 10.42006878, 21.09930847,
       20.77204   ,  9.38123953,  9.92874838, 13.87909156,  7.65989777,
       14.14644958, 23.12098439,  7.57859536,  4.46108779,  8.45288774,
        9.8332489 ,  8.79816602, 12.1844373 , 20.7286181 ,  8.85752264,
        9.77218289, 12.79519236, 15.46653997,  7.39548599, 16.07492022,
       11.39858287, 18.52903269, 14.78860526,  3.77843927, 14.77303586,
       15.40072018, 14.94191492, 20.69104132, 18.35259286, 17.31533647,
       12.16903971, 11.30678541, 17.06566779, 20.3674909 , 15.13059673,
       13.7812648 , 13.41202584, 12.05974864, 11.09426861, 14.30331278,
       16.46238457, 11.97603254,  8.43040344, 21.13549726,  8.76691178,
       18.70825024, 13.18304746, 16.36023345, 20.37940558, 19.39428859,
       17.05207517, 15.2345294 ,  9.00045731,  6.6571869 , 12.6032109 ,
       15.17576242,  9.0971137 , 19.18404789,  8.21235595, 10.2542979 ,
       17.45895403, 13.99099056, 14.29581352, 13.96499361,  9.98233931,
       12.41839002,  4.49899168, 13.14925984, 16.79434106,  7.14338905,
        5.76773362, 23.37216496, 16.2322095 , 18.02520375, 17.09364303,
        6.79455262, 11.67204805, 12.65937096, 12.77147555, 18.37379355,
```

```
 6.79455262, 11.67204805, 12.65937096, 12.77147555, 18.37379555,
 7.74741311, 19.79645703, 11.38055551, 14.96345873, 23.99282299,
15.67471434,  9.48739842, 10.73346659, 21.66789753, 10.29565275,
18.12550919, 14.6768742 , 10.34911491, 16.81840948, 17.21978367,
15.10871416, 20.39294059, 12.26061904, 10.65303081, 22.77515728,
18.01073434, 19.70800582, 13.43999429,  9.72012991, 13.92082105,
11.55954861, 11.65800904, 15.03697529, 13.90966708, 10.6082198 ,
16.42053875, 24.67716088,  8.14050662, 11.5834734 , 13.69358544,
14.33457738, 19.18786578,  6.61563167,  7.63409258,  6.41656236,
21.7803486 ,  4.50055193, 16.35248096, 17.30377429, 15.24161859,
 8.14650251, 15.36109512, 12.2070977 , 18.44176158, 19.63116108,
 6.03608517,  9.77943957, 23.14009955, 14.19726428, 12.11043732,
20.56739242, 11.63378238,  9.9151304 , 12.83834273,  6.55537142,
13.81233563, 10.43124495, 15.51465939,  9.72447624, 18.12895152,
24.01573225, 12.589141  , 17.33047014, 19.95792773, 10.13324681])
```

In [17]:

```
df = pd.DataFrame({'Actual':y_train, 'Predicted':y_pred})
df
```

Out[17]:

	Actual	Predicted
125	10.6	9.121747
68	18.9	18.977028
69	22.3	21.027629
108	5.3	3.602832
131	12.7	15.605008
...	...	...
98	25.4	24.015732
16	12.5	12.589141
74	17.0	17.330470
176	20.2	19.957928
82	11.3	10.133247

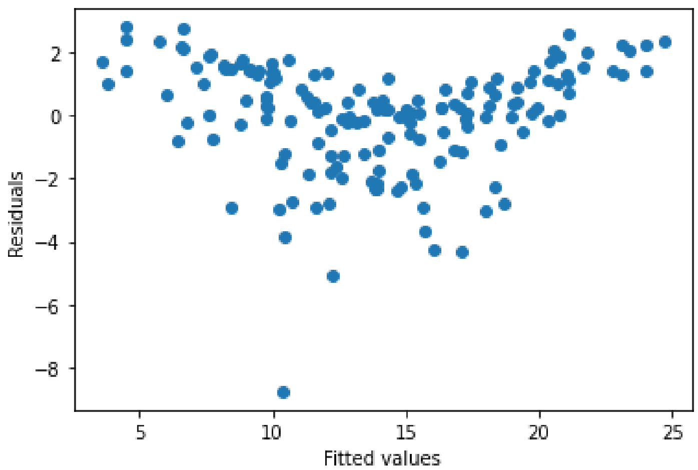
160 rows × 2 columns

In [18]:

```
plt.scatter(y_pred, (y_train-y_pred))
plt.xlabel("Fitted values")
plt.ylabel("Residuals")
```

Out[18]:

```
Text(0, 0.5, 'Residuals')
```

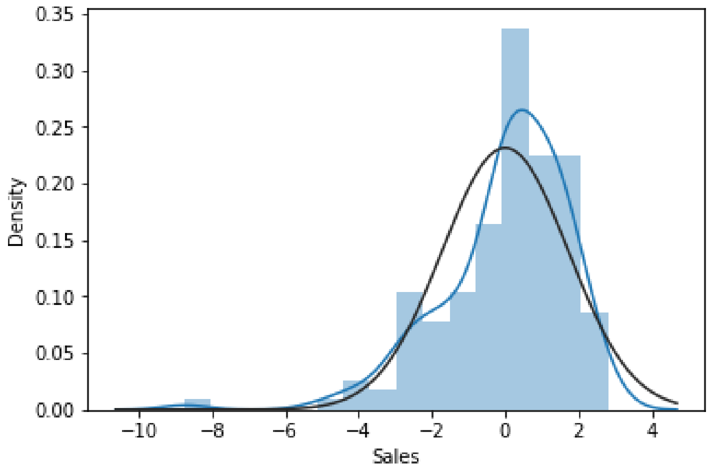


In [19]:

In [19]:

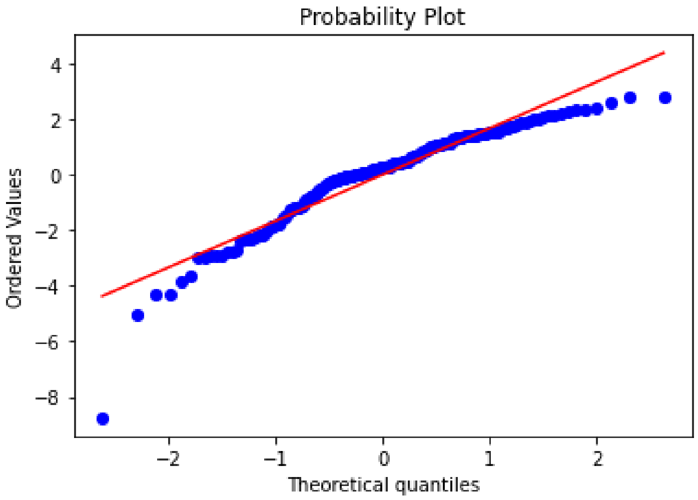
```
sns.distplot(y_train-y_pred, fit=norm)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [20]:

```
from scipy import stats
stats.probplot(y_train-y_pred, plot=plt)
plt.show()
```



In [21]:

```
import statsmodels.api as sm
x_endog = sm.add_constant(x_train)
x_endog1 = sm.add_constant(x_test)
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: p
andas.util.testing is deprecated. Use the functions in the public API at pandas.testing i
nstead.
  import pandas.util.testing as tm
```

In [22]:

```
res = sm.OLS(y_train, x_endog)
res.fit()
```

Out[22]:

```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f96c5422050>
```

In [23]:

```
res.fit().summary()
```

Out[23]:

OLS Regression Results

Dep. Variable:	Sales	**R-squared:**	0.887
Model:	OLS	**Adj. R-squared:**	0.885
Method:	Least Squares	**F-statistic:**	407.9
Date:	Fri, 10 Dec 2021	**Prob (F-statistic):**	1.37e-73
Time:	14:25:44	**Log-Likelihood:**	-314.00
No. Observations:	160	**AIC:**	636.0
Df Residuals:	156	**BIC:**	648.3
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	2.9965	0.375	7.983	0.000	2.255	3.738
TV	0.0460	0.002	28.540	0.000	0.043	0.049
Radio	0.1855	0.010	18.687	0.000	0.166	0.205
Newspaper	-0.0027	0.007	-0.378	0.706	-0.017	0.012

Omnibus:	51.704	**Durbin-Watson:**	2.274
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	127.325
Skew:	-1.360	**Prob(JB):**	2.25e-28
Kurtosis:	6.420	**Cond. No.**	465.

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

In [24]:

```
from sklearn import metrics
print("Mean Abolute Error for training data: ",metrics.mean_absolute_error(y_train, y_pred))
print("Mean Squared Error for training data: ",metrics.mean_squared_error(y_train, y_pred))
print("Root Mean Squared Error for training data: ",np.sqrt(metrics.mean_squared_error(y_train, y_pred)))
```

```
Mean Abolute Error for training data:  1.281994979746298
Mean Squared Error for training data:  2.96575198572951
Root Mean Squared Error for training data:  1.722135878997215
```

In [25]:

```
y_pred1 = regressor.predict(x_test)
print("Mean Abolute Error for testing data: ",metrics.mean_absolute_error(y_test, y_pred1))
print("Mean Squared Error for testing data: ",metrics.mean_squared_error(y_test, y_pred1))
print("Root Mean Squared Error for testing data: ",np.sqrt(metrics.mean_squared_error(y_test, y_pred1)))
```

```
Mean Abolute Error for testing data:  1.1800931227162388
Mean Squared Error for testing data:  2.088120709428994
Root Mean Squared Error for testing data:  1.445033117069984
```

In [119]:

```
def mean_absolute_percentage_error(y_true, y_pred):
```

```
def mean_absolute_percentage_error(y_true, y_pred):
  return np.mean(np.abs((y_true-y_pred)/y_true)*100)
```

In [27]:

```
print("Mean Absolute Percentage Error for training data: ",mean_absolute_percentage_error
(y_train, y_pred))
```

```
Mean Absolute Percentage Error for training data:  14.308336942801779
```

In [28]:

```
print("Mean Absolute Percentage Error for testing data: ",mean_absolute_percentage_error(
y_test, y_pred1))
```

```
Mean Absolute Percentage Error for testing data:  11.829477838487719
```

In [31]:

```
#Proceeding with an attempt to improve the efficiency by taking into account the statisti
cal interaction effects between independent variables!!!!!!!!!!!!!!!!!!
dataset
```

Out[31]:

	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	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	9.7
197	177.0	9.3	6.4	12.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	13.4

200 rows × 4 columns

In [64]:

```
TVRadio = []
for row in dataset.itertuples():
  TVRadio.append(row[1]*row[2])
dataset['TV*Radio'] = TVRadio
```

In [66]:

```
RadioNewspaper = []
for row in dataset.itertuples():
  RadioNewspaper.append(row[2]*row[3])
dataset['Radio*Newspaper'] = RadioNewspaper
```

In [67]:

```
TVNewspaper = []
for row in dataset.itertuples():
  TVNewspaper.append(row[1]*row[3])
dataset['TV*Newspaper'] = TVNewspaper
```

In [68]:

```
TVRadioNewspaper = []
```

```
for row in dataset.itertuples():
  TVRadioNewspaper.append(row[1]*row[2]*row[3])
dataset['TV*Radio*Newspaper'] = TVRadioNewspaper
```

In [69]:

```
dataset
```

Out[69]:

	TV	Radio	Newspaper	Sales	TV*Radio	Radio*Newspaper	TV*Newspaper	TV*Radio*Newspaper
0	230.1	37.8	69.2	22.1	8697.78	2615.76	15922.92	601886.376
1	44.5	39.3	45.1	10.4	1748.85	1772.43	2006.95	78873.135
2	17.2	45.9	69.3	9.3	789.48	3180.87	1191.96	54710.964
3	151.5	41.3	58.5	18.5	6256.95	2416.05	8862.75	366031.575
4	180.8	10.8	58.4	12.9	1952.64	630.72	10558.72	114034.176
...	...	...	...	...	...	...	...	...
195	38.2	3.7	13.8	7.6	141.34	51.06	527.16	1950.492
196	94.2	4.9	8.1	9.7	461.58	39.69	763.02	3738.798
197	177.0	9.3	6.4	12.8	1646.10	59.52	1132.80	10535.040
198	283.6	42.0	66.2	25.5	11911.20	2780.40	18774.32	788521.440
199	232.1	8.6	8.7	13.4	1996.06	74.82	2019.27	17365.722

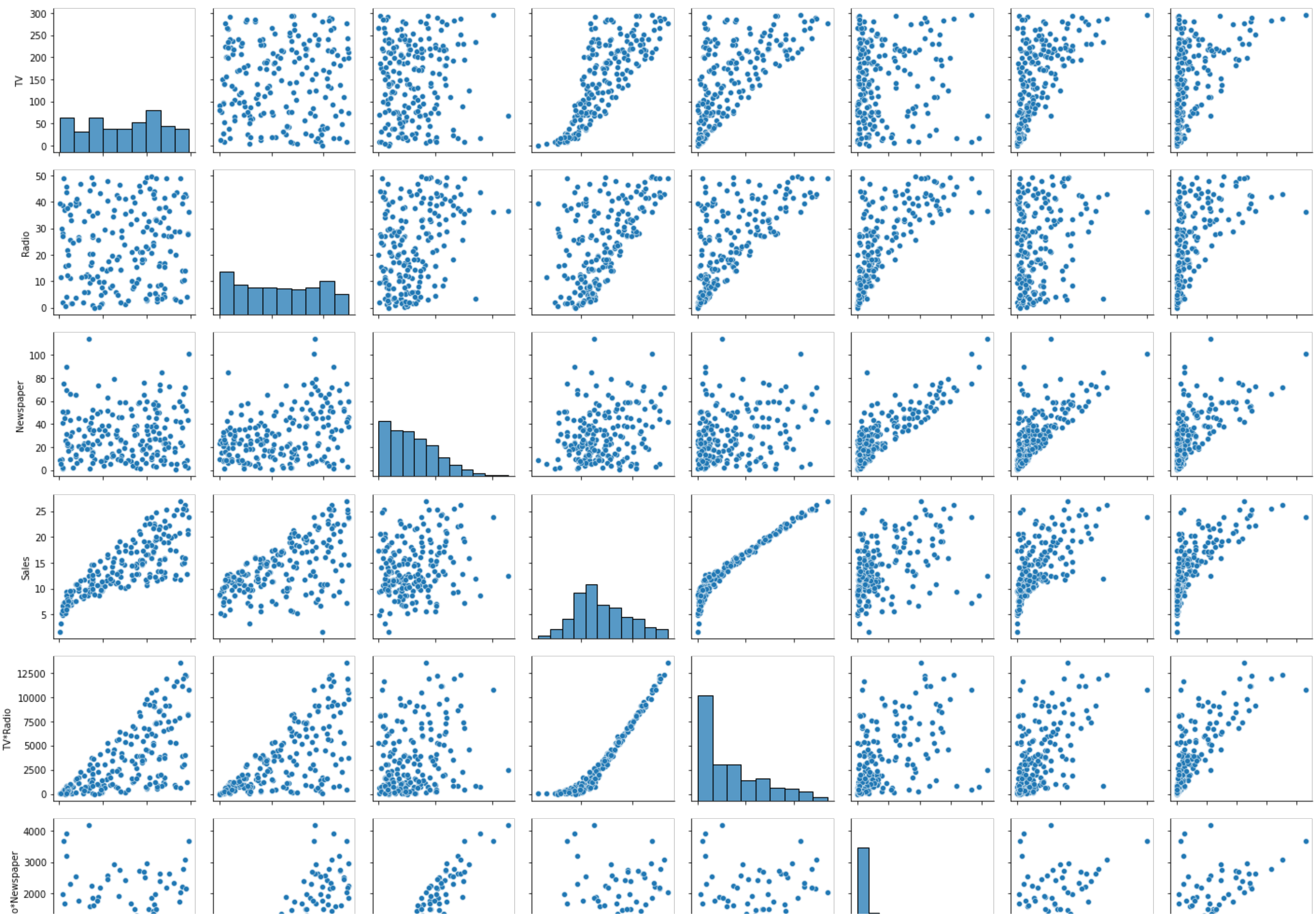
200 rows × 8 columns

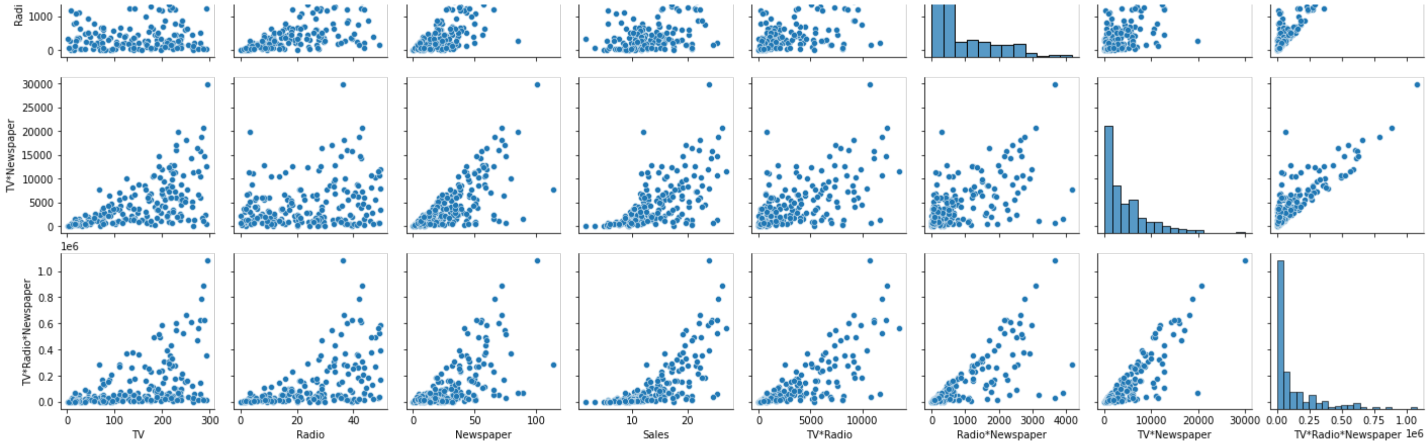
In [70]:

```
sns.pairplot(dataset)
```

Out[70]:

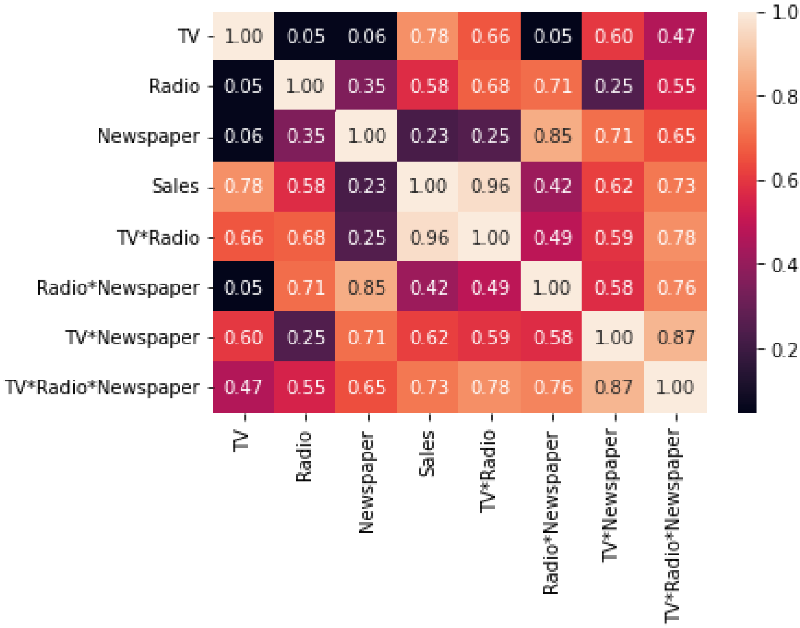
<seaborn.axisgrid.PairGrid at 0x7f96b270cd10>





In [72]:

```
sns.heatmap(dataset.corr(), annot=True, fmt=".2f")
plt.show()
```



In [102]:

```
x_sie = dataset[['TV', 'TV*Radio']]
y_sie = dataset['Sales']
```

In [103]:

```
x_train_sie, x_test_sie, y_train_sie, y_test_sie = train_test_split(x_sie, y_sie, test_s
ize=0.2, random_state=13)
```

In [104]:

```
regressor_sie = LinearRegression(fit_intercept = True)
regressor_sie.fit(x_train_sie, y_train_sie)
```

Out[104]:

```
LinearRegression()
```

In [105]:

```
coef_df_sie = pd.DataFrame(regressor_sie.coef_, x_sie.columns, columns = ['Coefficient_s
ie'])
coef_df_sie
```

Out[105]:

	Coefficient_sie
TV	0.015245

	Coefficient_sie
TV	0.015245
TV*Radio	0.001232

In [106]:

```
regressor_sie.intercept_
```

Out[106]:

```
7.508281984166364
```

In [107]:

```
y_pred_sie = regressor_sie.predict(x_train_sie)
y_pred_sie
```

Out[107]:

```
array([10.10535395, 19.17089146, 22.5395312 ,  7.71444135, 12.49869009,
       22.50893976, 19.81242892, 10.7289753 ,  8.90034705, 22.61623159,
       21.42628407, 10.22634943, 10.31101556, 12.07691675,  9.16968572,
       14.0912608 , 25.52757236,  8.2924604 ,  7.85737334,  9.66387244,
        9.58314849,  8.68569   ,  9.97436358, 22.02449853,  9.96982987,
        9.9755218 , 11.89163781, 13.19360236,  8.98510807, 12.51054365,
       11.18623561, 18.03783105, 12.61904752,  7.66163608, 14.35462604,
       15.13925894, 14.58175361, 22.07399258, 18.89289389, 17.41387142,
       11.73854543, 11.6497383 , 16.10868089, 20.83455328, 15.03164032,
       13.5764785 , 11.98576753, 12.12043812, 11.41759741, 14.25191961,
       16.35647456, 12.08282638,  7.87230123, 22.64630369,  9.83744347,
       16.76618177, 12.81928437, 16.1931422 , 21.57440144, 19.60831708,
       17.17887665, 13.50581946,  8.94321442,  8.73089317, 11.19417366,
       13.20947868,  9.97629014, 19.8891803 ,  9.51301469,  8.21856385,
       17.5268183 , 12.80366049, 12.30117747, 12.29292894, 10.60514423,
       10.11064227,  7.99713123, 12.67027874, 15.72622408,  8.91980105,
        8.45569557, 25.98019705, 15.39466864, 18.29130644, 13.40833176,
        8.18645986,  9.82568936, 11.31725729, 12.80649486, 16.59461929,
        8.29751374, 20.68580958,  9.12863898, 14.84535845, 27.12935161,
       12.64975624, 10.1646053 ,  8.61039035, 23.43173235,  9.00647662,
       18.37840996, 12.37526491,  7.55310609, 16.51554663, 17.49030537,
       14.83338013, 21.7323077 ,  8.16506823, 10.39722411, 25.23849254,
       15.55523806, 19.86636467, 12.59479779,  9.53595126, 13.52045162,
       11.25662699, 10.70852334, 14.63258301, 11.04151114, 11.11665106,
       16.02331072, 28.41220293,  9.49251782, 11.62856062, 11.58541633,
       14.32797968, 19.93111845,  8.74826798,  9.13225588,  7.96809627,
       23.6039413 ,  7.98171753, 16.15901179, 17.1298842 , 13.97231132,
        9.42516515, 13.32797649, 10.34136629, 18.68244758, 20.32782987,
        8.07213478,  9.61060759, 25.89806378, 13.98478865,  8.74318122,
       21.92523516,  8.67583398, 10.38967297, 12.23467129,  8.75970527,
       11.83030595,  8.00102349, 15.25337395,  9.99582814, 18.49848626,
       27.02276975, 11.59921406, 17.22937981, 20.53760558, 10.53952238])
```

In [108]:

```
df_sie = pd.DataFrame({'Actual':y_train_sie, 'Predicted':y_pred_sie})
df_sie
```

Out[108]:

	Actual	Predicted
125	10.6	10.105354
68	18.9	19.170891
69	22.3	22.539531
108	5.3	7.714441
131	12.7	12.498690
...	...	...
98	25.4	27.022770
16	12.5	11.599214

	Actual	Predicted
74	17.0	17.229380
176	20.2	20.537606
82	11.3	10.539522

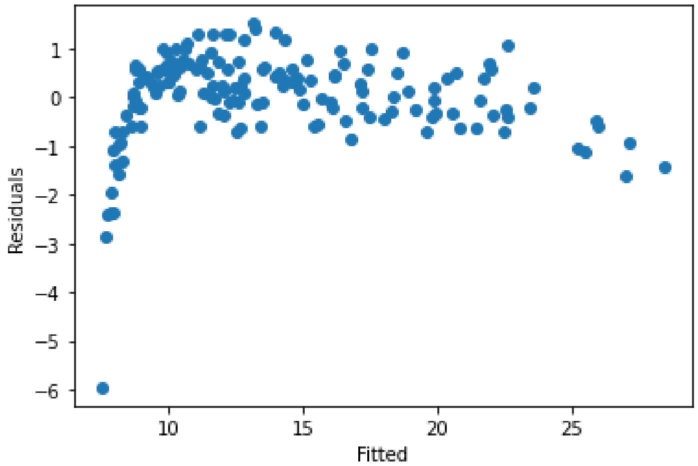
160 rows × 2 columns

In [109]:

```
plt.scatter(y_pred_sie, (y_train_sie-y_pred_sie))
plt.xlabel('Fitted')
plt.ylabel('Residuals')
```

Out[109]:

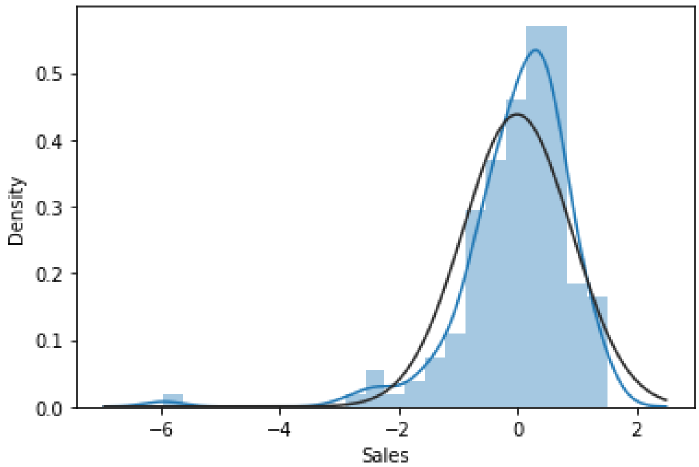
```
Text(0, 0.5, 'Residuals')
```



In [110]:

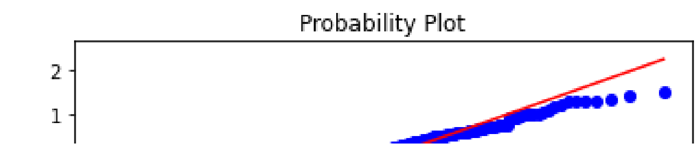
```
sns.distplot((y_train_sie-y_pred_sie), fit=norm)
plt.show()
```

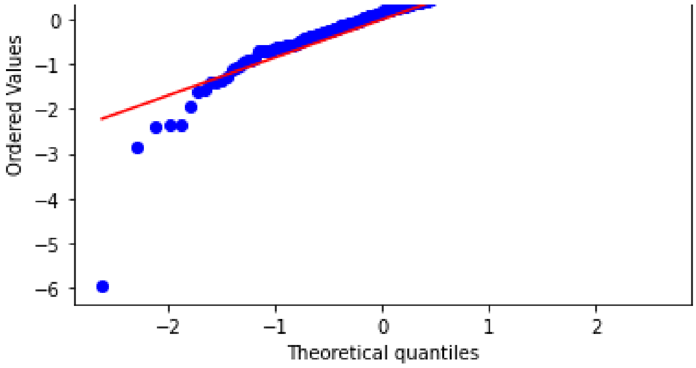
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [111]:

```
from scipy import stats
stats.probplot((y_train_sie-y_pred_sie), plot=plt)
plt.show()
```





In [112]:

```
import statsmodels.api as sm
x_endog_sie = sm.add_constant(x_train_sie)
x_endog_sie1 = sm.add_constant(x_test_sie)
```

In [113]:

```
res_sie = sm.OLS(y_train_sie, x_endog_sie)
res_sie.fit()
```

Out[113]:

```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f96affb8b50>
```

In [114]:

```
res_sie.fit().summary()
```

Out[114]:

OLS Regression Results

Dep. Variable:	Sales	**R-squared:**	0.968
Model:	OLS	**Adj. R-squared:**	0.968
Method:	Least Squares	**F-statistic:**	2403.
Date:	Fri, 10 Dec 2021	**Prob (F-statistic):**	1.85e-118
Time:	16:07:55	**Log-Likelihood:**	-212.11
No. Observations:	160	**AIC:**	430.2
Df Residuals:	157	**BIC:**	439.4
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	7.5083	0.142	52.819	0.000	7.228	7.789
TV	0.0152	0.001	13.629	0.000	0.013	0.017
TV*Radio	0.0012	2.89e-05	42.643	0.000	0.001	0.001

Omnibus:	104.807	**Durbin-Watson:**	2.078
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	906.206
Skew:	-2.272	**Prob(JB):**	1.66e-197
Kurtosis:	13.737	**Cond. No.**	9.30e+03

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 9.3e+03. This might indicate that there are strong multicollinearity or other numerical problems.

In [115]:

```
print("Mean Absolute Error for Training Data:",metrics.mean_absolute_error(y_train_sie,y_pred_sie))
print("Mean Squared Error for Training Data:",metrics.mean_squared_error(y_train_sie,y_pred_sie))
print("Root Mean Squared Error for Training Data:",np.sqrt(metrics.mean_squared_error(y_train_sie,y_pred_sie)))
```

```
Mean Absolute Error for Training Data: 0.6372741758347489
Mean Squared Error for Training Data: 0.8298183589582951
Root Mean Squared Error for Training Data: 0.9109436639871289
```

In [117]:

```
y_pred_sie1 = regressor_sie.predict(x_test_sie)

print("Mean Absolute Error for Testing Data:",metrics.mean_absolute_error(y_test_sie,y_pred_sie1))
print("Mean Squared Error for Testing Data:",metrics.mean_squared_error(y_test_sie,y_pred_sie1))
print("Root Mean Squared Error for Testing Data:",np.sqrt(metrics.mean_squared_error(y_test_sie,y_pred_sie1)))
```

```
Mean Absolute Error for Testing Data: 0.7905602081155837
Mean Squared Error for Testing Data: 1.2850818982455423
Root Mean Squared Error for Testing Data: 1.1336145280674301
```

In [120]:

```
print("Mean Absolute Percentage Error For Training Data:",mean_absolute_percentage_error(y_train_sie, y_pred_sie))
```

```
Mean Absolute Percentage Error For Training Data: 7.941941263501843
```

In [122]:

```
print("Mean Absolute Percentage Error For Testing Data:",mean_absolute_percentage_error(y_test, y_pred_sie1))
```

```
Mean Absolute Percentage Error For Testing Data: 9.412022462215504
```

In [ ]:

```
```