```python
In [178]:
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import Lasso
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.metrics import accuracy_score
```

```python
In [179]:
df=pd.read_csv('https://raw.githubusercontent.com/dsrscientist/DSData/master/Advertising.csv')
```

```python
In [180]:
df
```

Out[180]:

	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

```python
In [181]:
type(df)
Out[181]:
pandas.core.frame.DataFrame
```

```python
In [182]:
df.isnull().sum()
Out[182]:
Unnamed: 0    0
TV            0
radio         0
newspaper     0
sales         0
dtype: int64
```

- There is no null values present in data set

```python
In [183]:
df.dtypes
Out[183]:
Unnamed: 0      int64
TV            float64
radio         float64
newspaper     float64
sales         float64
dtype: object
```

```python
In [184]:
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  200 non-null    int64
 1   TV          200 non-null    float64
 2   radio       200 non-null    float64
 3   newspaper   200 non-null    float64
 4   sales       200 non-null    float64
dtypes: float64(4), int64(1)
memory usage: 7.9 KB
```

```python
In [185]:
#https://www.abacademies.org/articles/prediction-of-sales-based-on-an-effective-advertising-media-sale-data-a-python-implementation-approach-11437.html
#https://github.com/SidSolanki28/Predict-Sales-by-Advertising-Ads/blob/master/Advertising_Ads.ipynb
```

```python
In [186]:
df['sales'].nunique()
Out[186]:
121
```

```python
In [187]:
df.shape[0]
Out[187]:
200
```

```python
In [188]:
df.columns
Out[188]:
Index(['Unnamed: 0', 'TV', 'radio', 'newspaper', 'sales'], dtype='object')
```

```python
In [189]:
dt=df[['TV', 'radio', 'newspaper', 'sales']].copy()
```

```python
In [190]:
dt.columns
Out[190]:
Index(['TV', 'radio', 'newspaper', 'sales'], dtype='object')
```

```python
In [191]:
dt.describe()
```

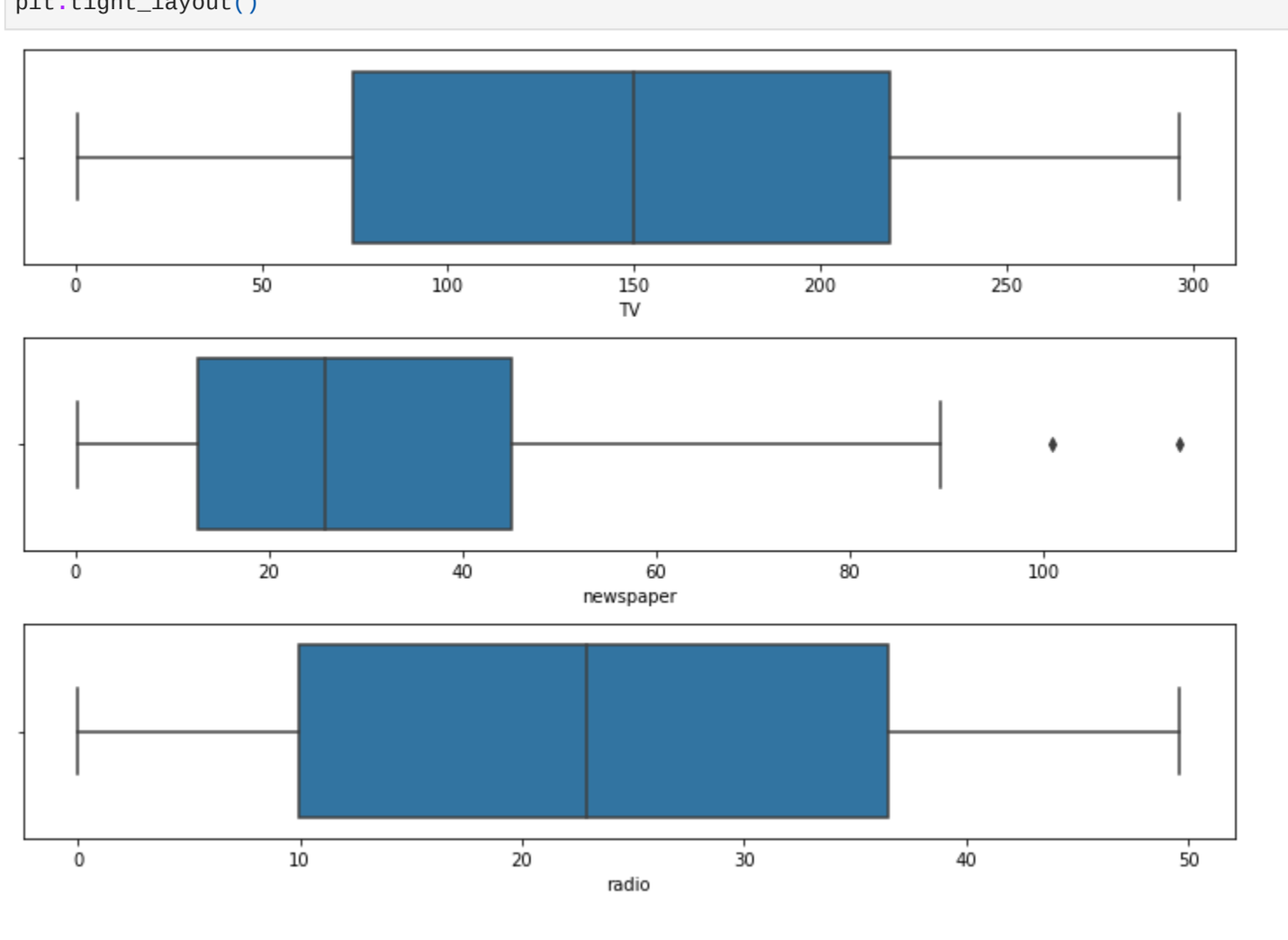
Out[191]:

	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

- From the grapg there is more spending on TV advertisement which is 147,000 and less on radio of 23,000

Outlier Checking

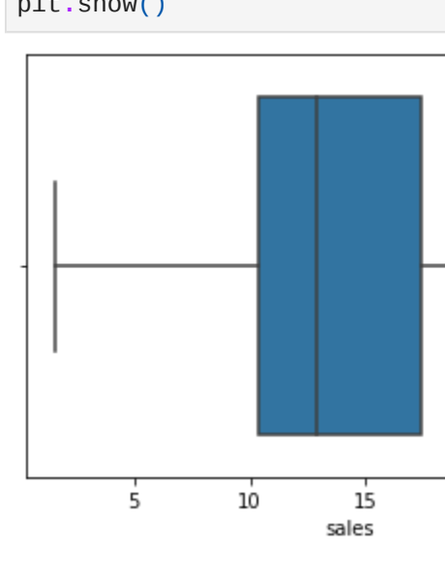
```python
In [192]:
fig, axs = plt.subplots(3, figsize = (10,7))
pt1 = sns.boxplot(dt['TV'], ax = axs[0])
pt2 = sns.boxplot(dt['newspaper'], ax = axs[1])
pt3 = sns.boxplot(dt['radio'], ax = axs[2])
plt.tight_layout()
```



- There are no considerable outliers present in data

Data Visualization

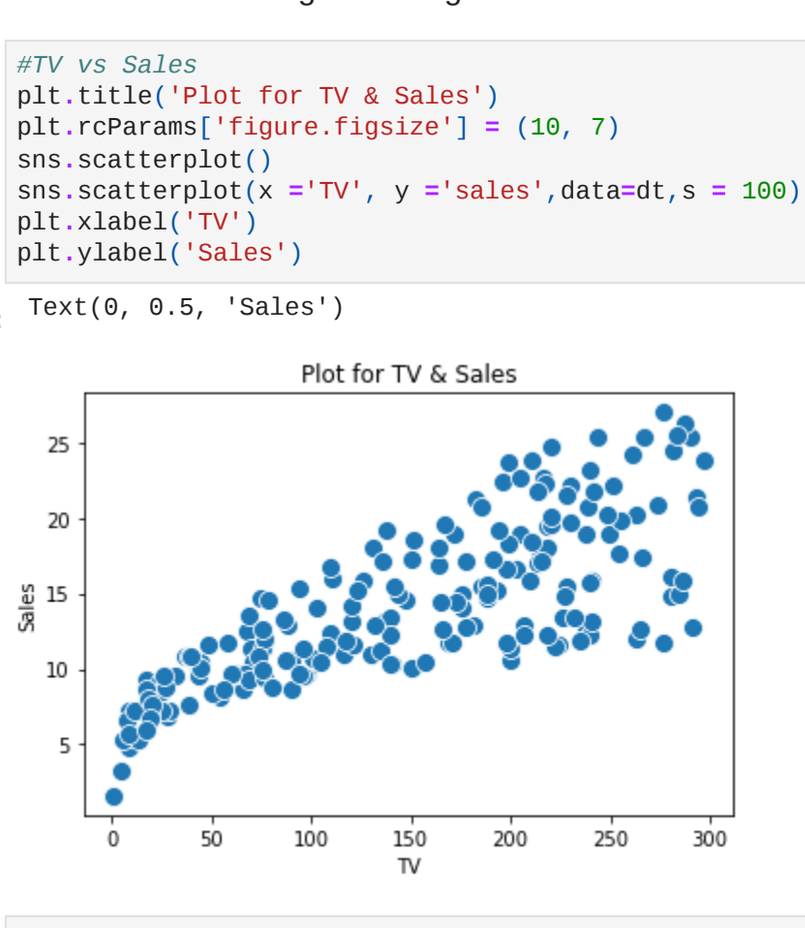
```python
In [193]:
sns.boxplot(dt['sales'])
plt.show()
```



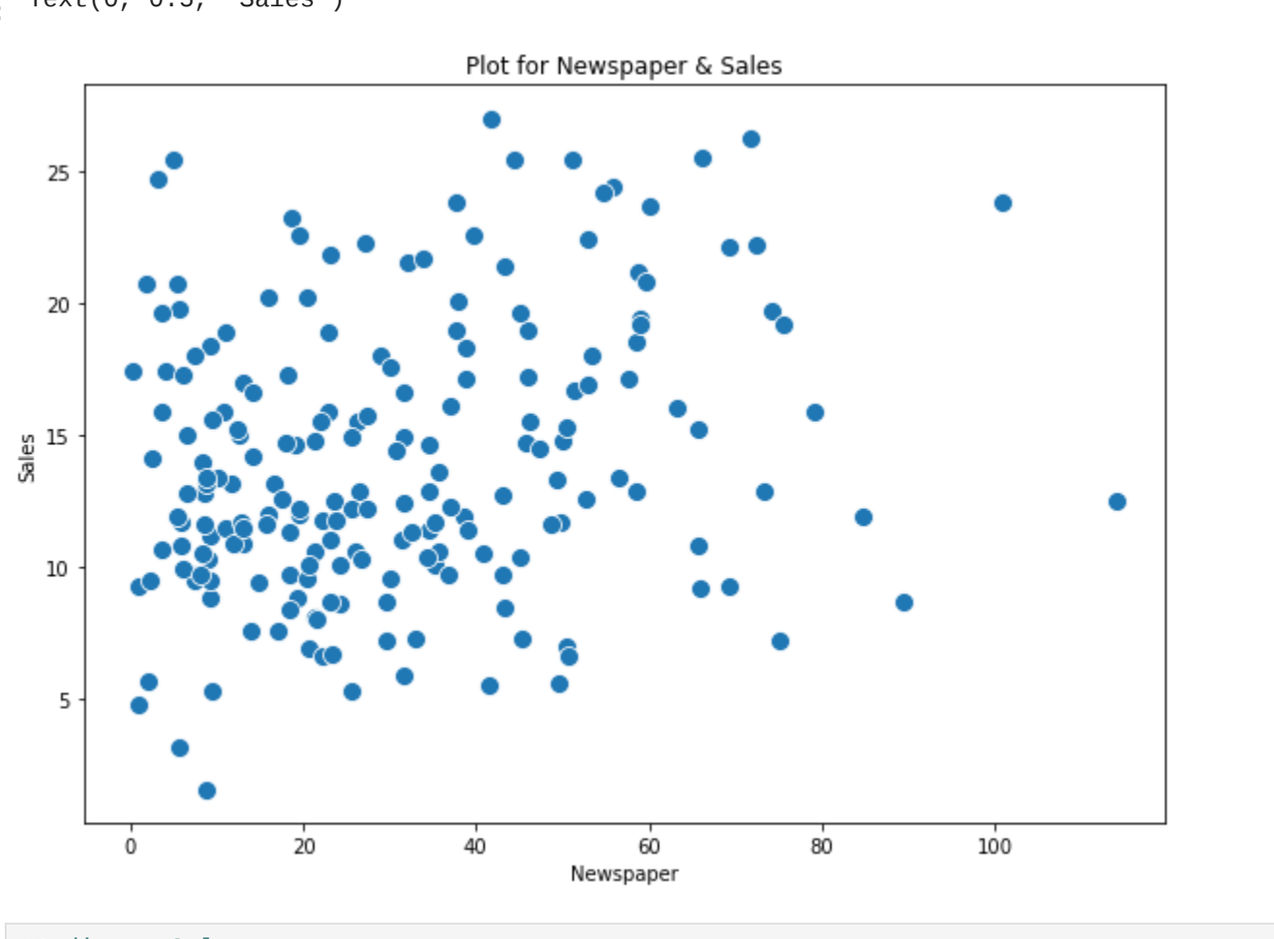
- No outlier present in target Sales
- There is average sales of about 15,000

Scatter Plot Showing How Target Variable Sales is Related to Other Variables

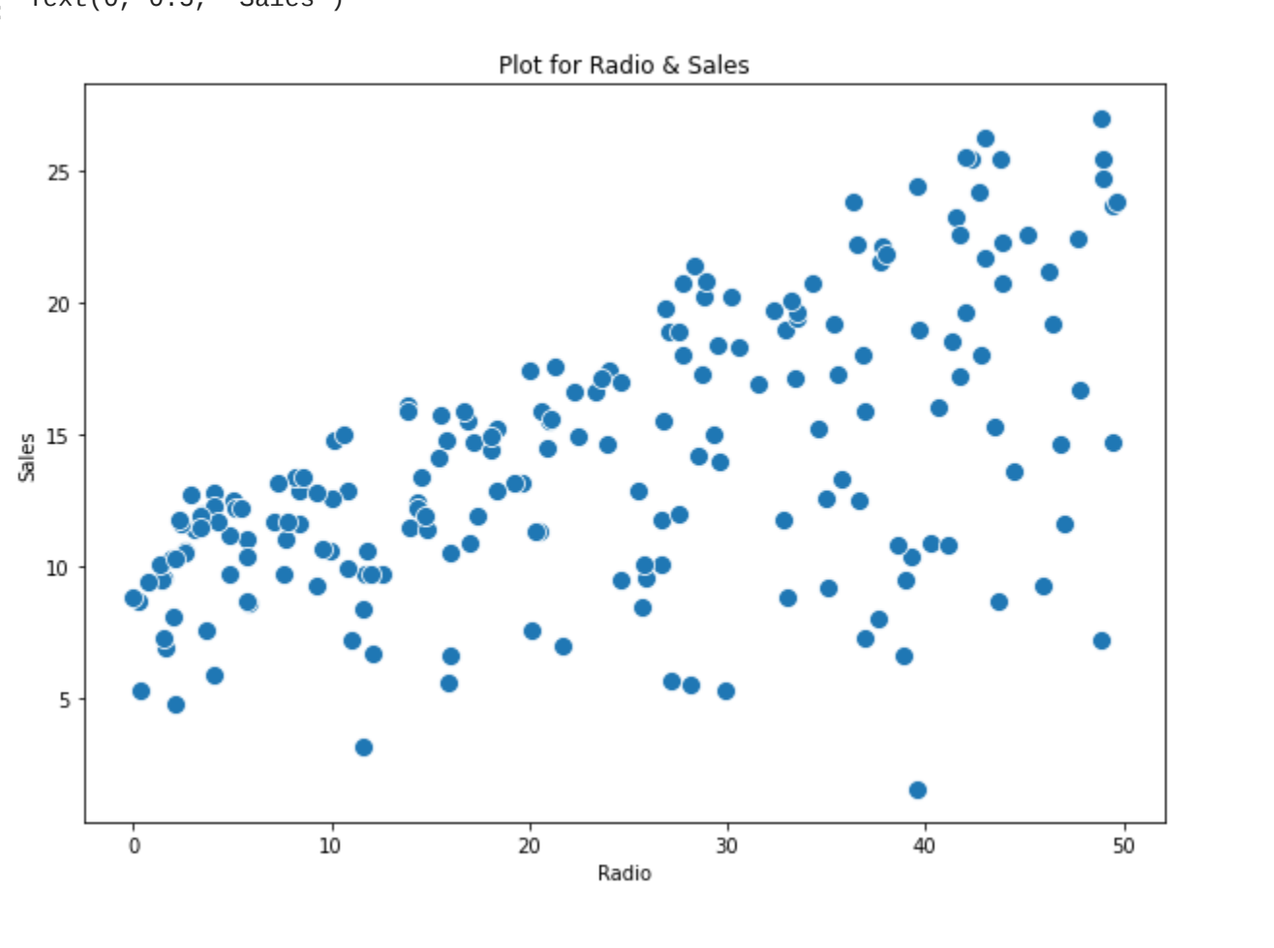
```python
In [194]:
#TV vs Sales
plt.title('Plot for TV & Sales')
plt.rcParams['figure.figsize'] = (10, 7)
sns.scatterplot()
sns.scatterplot(x ='TV', y ='sales',data=dt,s = 100)
plt.xlabel('TV')
plt.ylabel('Sales')
Out[194]:
Text(0, 0.5, 'Sales')
```




```python
In [195]:
#Newspaper vs Sales
plt.title('Plot for Newspaper & Sales')
plt.rcParams['figure.figsize'] = (10,7)
sns.scatterplot()
sns.scatterplot(x ='newspaper', y ='sales',data=dt, s=100)
plt.xlabel('Newspaper')
plt.ylabel('Sales')
Out[195]:
Text(0, 0.5, 'Sales')
```

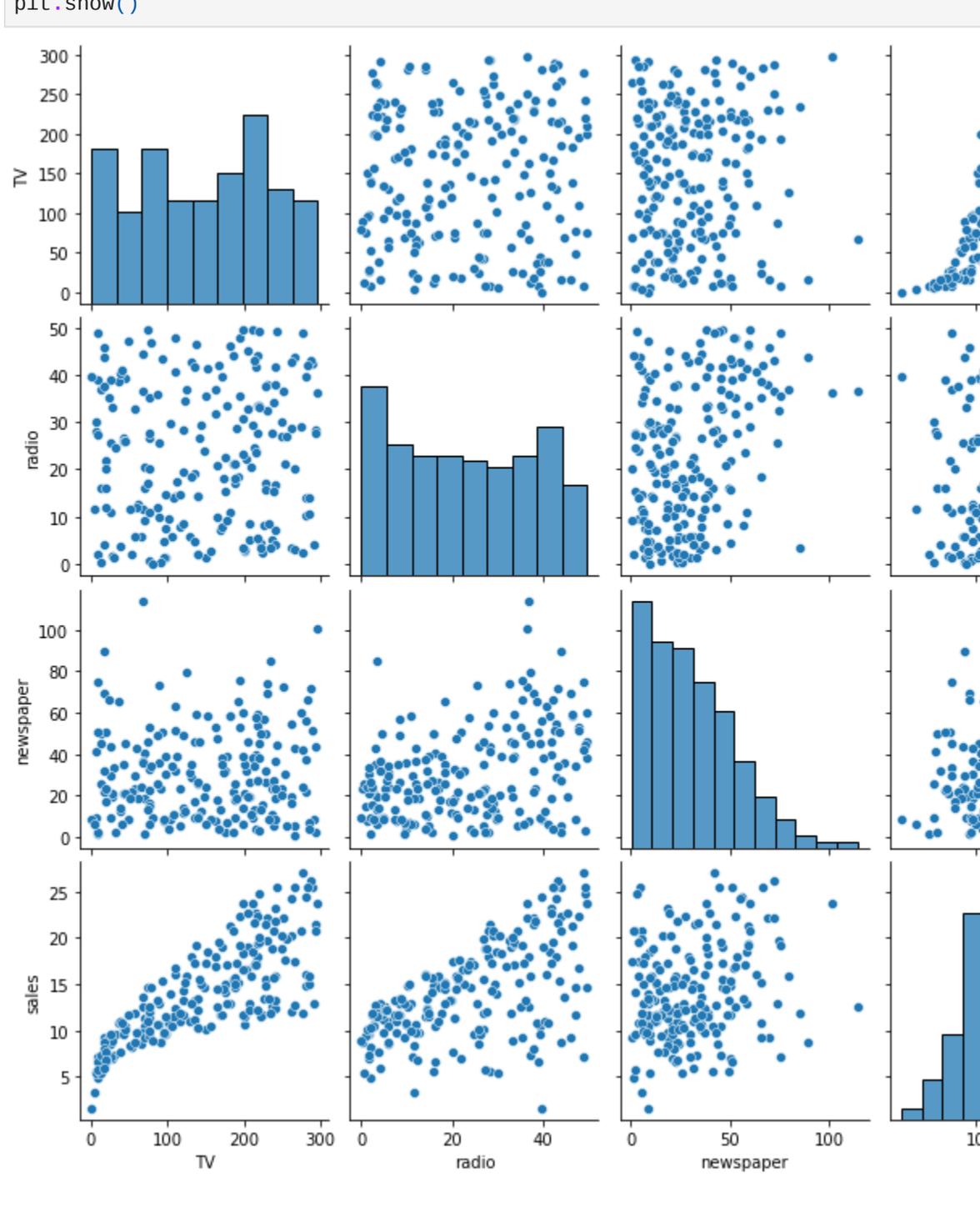



```python
In [196]:
#Radio vs Sales
plt.title('Plot for Radio & Sales')
plt.rcParams['figure.figsize'] = (10,7)
sns.scatterplot()
sns.scatterplot(x ='radio', y ='sales',data=dt, s=100)
plt.xlabel('Radio')
plt.ylabel('Sales')
Out[196]:
Text(0, 0.5, 'Sales')
```



- from the above figures, it shows that TV data set seems to be more linear as compared to other variable dispersion of values

```python
In [197]:
sns.pairplot(dt, size=2.5)
plt.show()
```



- Target Sales is directly correlated with TV, bit with Radio but no correlation with Newspaper present here. also there is no correlation seems between all features.
- Spending budget for advertisments on TV is in all range, same for Radio but less than TV. But on newspapers, less budget is spend

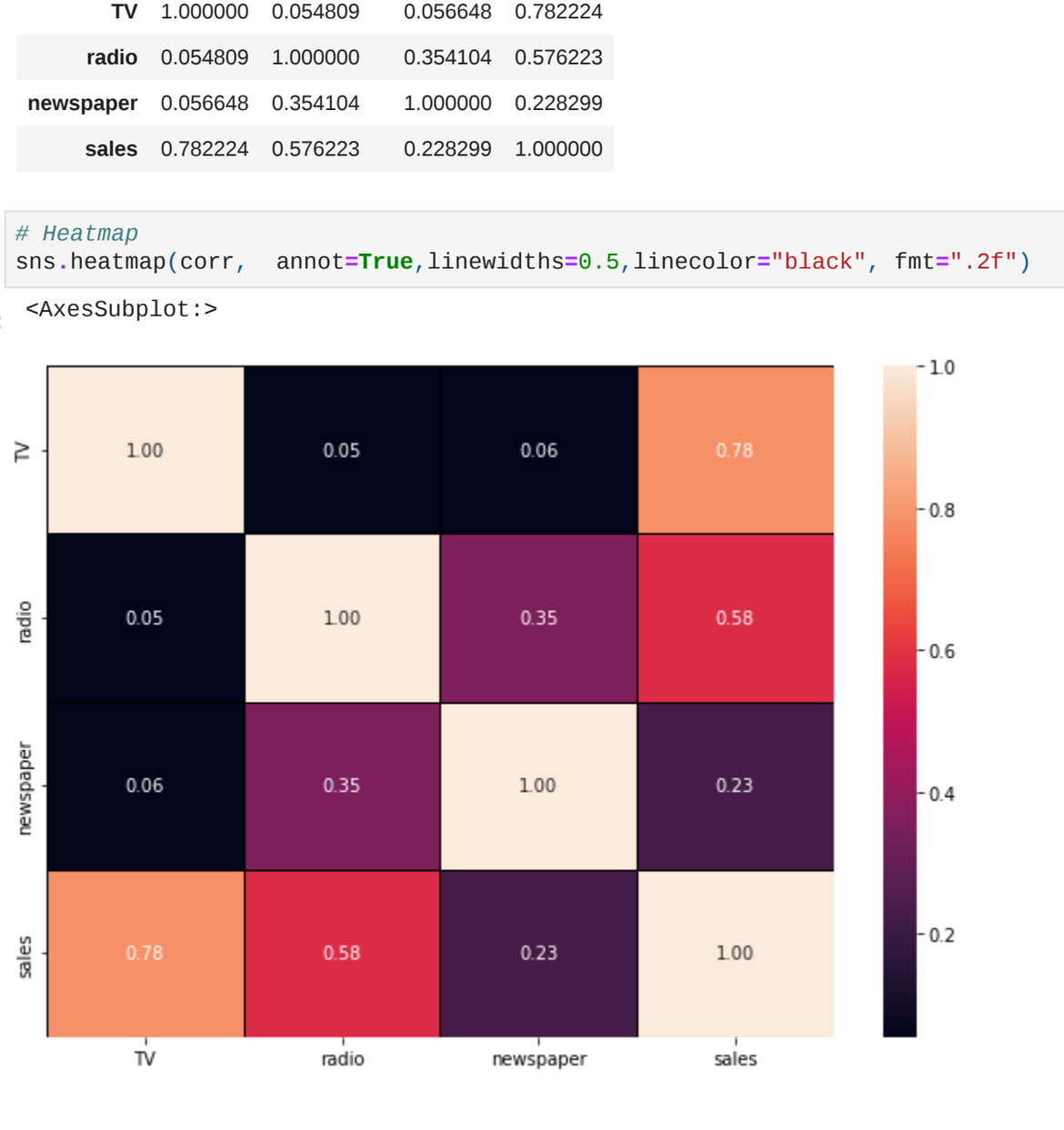
Correlation

```python
In [198]:
# Calculate correlations
corr = dt.corr()
corr
```

Out[198]:

	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

```python
In [199]:
# Heatmap
sns.heatmap(corr,  annot=True,linewidths=0.5,linecolor="black", fmt=".2f")
Out[199]:
<AxesSubplot:>
```



- All the columns of the data set is positively correlated with the target column
- It is visible from the pairplot and the heatmap, the correlation factor for variable TV is 0.78 seems to be most correlated with Sales

```python
In [200]:
dt.skew()
Out[200]:
TV          -0.069853
radio        0.094175
newspaper    0.894720
sales        0.407571
dtype: float64
```

```python
In [ ]:
```

Data Preprocessing

```python
In [201]:
X = dt.iloc[:, :-1].values
y = dt.iloc[:, -1].values
```

```python
In [202]:
from sklearn.preprocessing import power_transform

X=power_transform(X,method='yeo-johnson')
```

```python
In [203]:
type(X)
Out[203]:
numpy.ndarray
```

```python
In [204]:
X
Out[204]:
array([[ 0.94867429,  0.96224689,  1.51433531],
       [-1.19131426,  1.0401788 ,  0.78768252],
       [-1.6477566 ,  1.37070964,  1.51699753],
       [ 0.14102023,  1.14238689,  1.21465643],
       [ 0.45271493, -0.74865064,  1.21170398],
       [-1.82382233,  1.51501853,  1.66502354],
       [-1.00249116,  0.69372704, -0.1077535 ],
       [-0.210275  , -0.10158544, -0.84409341],
       [-1.82609501, -1.63345378, -2.13259669],
       [ 0.64764492, -1.56676988, -0.22900395],
       [-0.88377378, -1.20298758, -0.07226156],
       [ 0.79708355,  0.18072579, -1.60261898],
       [-1.52641627,  0.81900563,  1.42512925],
       [-0.48036844, -1.02813642, -1.22982506],
       [ 0.69106577,  0.6992403 ,  0.81860579],
       [ 0.60295713,  1.45770359,  1.04432344],
       [-0.86079972,  0.89906999,  2.52943514],
       [ 1.4355201 ,  1.05563142,  1.13383286],
       [-0.84199328, -0.04209908, -0.39356015],
       [ 0.09508551,  0.17453165, -0.34671544],
       [ 0.8337607 ,  0.40367061,  1.05996548],
       [ 1.01958629, -1.27567304, -0.1077535 ],
       [-1.72701642, -0.35739289,  0.93875739],
       [ 0.93109798, -0.28631129,  0.02589311],
       [-0.93569772, -0.60431061, -0.39356015],
       [ 1.26290943, -1.4550585 , -0.3237259 ],
       [ 0.04659013,  0.49664406, -0.76867281],
       [ 1.0456669 , -0.30040383, -0.1386789 ],
       [ 1.12918152,  0.3682981 , -0.1386789 ],
       [-0.82328646, -0.35021379,  0.63452018],
       [ 1.54129731,  0.43876195,  0.72114861],
       [-0.29553337, -0.25134036,  0.55235877],
       [-0.48404243, -1.71918898,  0.20071936],
       [ 1.28829734, -0.07502706, -2.32472495],
       [-0.50245749, -1.7342107 , -1.20981332],
       [ 1.52114819, -1.3853128 , -1.10471076],
       [ 1.3004967 ,  1.2675887 , -1.47330873],
       [-0.76905257,  1.53874417,  0.80833914],
       [-1.21245403,  0.34455612,  0.4155984 ],
       [ 0.92816503,  0.95701091,  0.28742321],
       [ 0.67493776,  0.07409584,  0.27034854],
       [ 0.41309098,  0.72671408,  0.55615392],
       [ 1.54769995,  0.40367061, -1.95904851],
       [ 0.71920943, -0.95499286,  0.03545918],
       [-1.50359538,  0.28462118,  0.72469411],
       [ 0.39319467,  0.08679116,  0.26605961],
       [-0.57689562, -0.82396106,  0.43960934],
       [ 1.04373768,  1.15250515, -0.3817378 ],
       [ 0.92033882, -0.36458821,  0.94852755],
       [-0.87294338, -0.67550361,  0.48300508],
       [ 0.64764492, -1.50354208,  0.3953996 ],
       [-0.44500506, -0.84958728, -1.65883577],
       [ 0.81395585,  1.16260524,  0.59004665],
       [ 0.47140703,  1.38529598,  1.22055211],
       [ 1.2610261 ,  0.46779569, -0.54176985],
       [ 0.63852563,  1.53874417,  1.25857888],
       [-1.85624738,  0.42709558,  0.65646046],
       [-0.02802214, -0.12834141, -0.49726342],
       [ 0.75824197,  1.54820925,  0.51793191],
       [ 0.75724354,  0.50813245, -1.03293213],
       [-1.05927427, -1.64727037, -0.21815975],
       [ 1.24783212,  1.21283741,  1.10022212],
       [ 1.03794746, -0.38627294,  0.07797929],
       [-0.41714955,  0.51386586, -1.11394312],
       [-0.08546826,  1.21783637,  0.15154706],
       [-0.84467375, -0.87549189, -2.1571572 ],
       [-1.39531449,  0.21769395, -1.88378271],
       [ 0.0066178 , -0.45966823, -0.95619942],
       [ 1.01958629,  0.39191142, -0.8911416 ],
       [ 0.81792069,  1.2725405 ,  0.07329661],
       [ 0.64055308,  0.5708121 ,  0.55615392],
       [-0.3321763 , -0.4745618 ,  0.27462933],
       [-1.47420382,  0.70474737, -0.33518557],
       [-0.10474749, -1.21318999,  0.25745713],
       [ 0.7841573 ,  0.21769395, -0.73224866],
       [-1.65352738,  1.26263268,  2.01064695],
       [-1.46224158, -1.70439494, -0.25638211],
       [-0.20680098,  0.45039797, -0.65484297],
       [-1.90271957,  0.53102334, -1.02420602],
       [-0.25915513, -1.01885558, -0.1283175 ],
       [-0.74679627,  0.34455612, -0.17008941],
       [ 1.04277291, -1.3853128 ,  0.48691113],
       [-0.76118235, -0.05523482,  0.30858709],
       [-0.85272745,  1.30216339,  0.43562451],
       [ 0.78515237,  1.22782124,  0.36271243],
       [ 0.58051341, -0.1824681 ,  1.41963599],
       [-0.74810183,  0.39191142, -0.53534269],
       [-0.32151022,  1.10682827,  1.35009138],
       [-0.59445094,  0.27253204,  1.62418784],
       [-0.3321763 ,  1.46250027,  0.99685054],
       [-0.04935604, -1.29701657, -1.03293213],
       [-1.44360046, -1.71918898,  0.32955606],
       [ 0.8268345 ,  0.7321905 ,  1.22937273],
       [ 1.14922339,  0.89377076,  1.59579653],
       [-0.36073275, -0.49704225, -0.89912378],
       [ 0.26831445,  0.62707446,  1.04432344],
       [ 0.62533398, -1.4550585 , -1.3679351 ],
       [ 0.49522053, -0.00946213, -0.18598289],
       [ 1.51197614,  1.19279775,  0.99045745],
       [-0.03924063,  1.16260524,  0.8151881 ],
       [ 0.87322688, -1.36277082,  0.94527481],
       [ 1.57327016,  0.88315609,  2.2628248 ],
       [ 1.42441703, -0.80702634, -0.21815975],
       [ 0.52616407, -0.26528466, -0.41743473],
       [ 1.02732204,  0.77578583, -1.4371576 ],
       [-0.00900053,  1.39500055,  1.22937273],
       [-1.50533991, -0.73221674,  0.18741746],
       [-0.56814506, -1.91962741, -0.12315678],
       [-1.72906885, -1.90076688, -0.00306944],
       [ 1.19202117,  0.35644335, -1.41364623],
       [ 0.90662534, -0.97304835,  1.15501188],
       [ 1.06108536,  0.97270342, -0.12315678],
       [ 0.39948386, -0.39353456, -1.84832983],
       [ 0.74625251, -0.0355487 , -0.91521378],
       [-0.72337219,  1.41436293,  0.39133878],
       [-0.76380392,  0.81362329,  1.03804144],
       [ 0.00550359, -0.4745618 , -0.00306944],
       [-0.74679627, -1.82997283, -0.6140794 ],
       [-0.1469415 ,  0.91493549,  1.76974284],
       [-1.60617806, -0.35021379, -0.17008941],
       [ 0.02885793,  0.35050382,  0.82542768],
       [-1.61739376,  0.0357579 ,  0.96473213],
       [ 0.88896054, -1.59299019, -0.56119394],
       [-0.17678774,  0.79203663, -0.78347482],
       [ 0.94281959,  0.66606674,  1.64467302],
       [-0.60829585, -0.66750027,  0.01146189],
       [-1.84451116,  1.01950683,  0.9711866 ],
       [-0.69751107, -1.9798635 , -1.04171092],
       [ 0.85253082,  1.51977089, -1.71817136],
       [-0.9730961 , -0.65156482,  0.71759813],
       [-2.04026249,  1.05563142, -1.08642518],
       [ 1.2845405 , -1.52845902,  0.71404265],
       [-1.83065941,  0.37421339, -1.90202575],
       [ 0.84759549,  0.7321905 ,  0.78768252],
       [-1.30833884,  1.0039509 ,  1.41688552],
       [-1.1347862 ,  1.42402092, -1.10471076],
       [-1.49489926,  1.02468221, -1.03293213],
       [ 1.36405418,  0.47358005,  1.2498484 ],
       [-1.2139708 ,  0.2966758 , -0.26744284],
       [ 0.49522053,  1.2725405 , -1.97889407],
       [-0.78616198, -0.27928757, -0.74672069],
       [ 0.58562018,  0.83511859,  1.68019945],
       [ 0.85450412,  0.71574298,  0.52562487],
       [-0.39426242, -1.21318999,  0.38727085],
       [-0.49631074, -0.43746469,  0.56372647],
       [ 0.01774834, -1.66126392, -1.05942991],
       [ 1.0456669 , -1.05622936, -1.08642518],
       [ 1.07551559,  1.51977089,  0.75987942],
       [-1.29103896,  1.09151903, -0.82108927],
       [-1.18830853,  0.29065278, -0.26190454],
       [ 1.42904484, -0.50457379,  0.49081078],
       [-0.20101603, -0.95499286,  0.90922967],
       [ 0.62533398,  0.13716541, -0.65484297],
       [ 0.35322947,  1.0607726 ,  0.51793191],
       [ 0.52513474, -0.00296885, -1.01553176],
       [-1.93665721, -0.68353104, -1.39057916],
       [-0.52465633,  1.25270792,  0.96796131],
       [ 0.12246872, -1.74947332, -0.06724139],
       [-1.75822712,  0.91493549,  0.79113681],
       [-0.07867963, -0.1824681 ,  0.3953996 ],
       [ 0.36587519, -0.20298441,  0.23144984],
       [-0.62724967,  0.85652391,  0.92895126],
       [ 0.53130856, -0.20298441, -0.00306944],
       [ 0.27045056,  0.90965233, -1.20981332],
       [-0.24514238, -0.44484785, -1.42534498],
       [ 0.99148599, -1.46701964,  1.90420913],
       [-1.63438885,  0.95176976, -0.20737541],
       [ 0.71820605, -1.26510158, -0.32944703],
       [ 0.80403524,  0.15589202,  1.18797267],
       [ 1.46230082, -0.76519122, -1.31304986],
       [-1.10987617, -0.68353104, -0.38763953],
       [ 0.28112073, -0.01596668,  0.86598739],
       [-1.60245911, -0.06841773, -0.47232241],
       [ 0.32257166, -1.07517365, -0.75400514],
       [ 0.87322688, -1.46701964, -0.73224866],
       [ 1.39381826,  1.51501853,  0.67098058],
       [ 1.12535894,  0.54811736, -0.27856754],
       [ 0.34161695, -1.00961537,  0.41961731],
       [ 1.39196069, -1.60632069, -0.09754949],
       [ 0.29283811, -0.81547901, -0.43554699],
       [ 0.19634785, -1.56676988, -1.12323656],
       [ 0.83474968, -1.2441519 ,  0.08265171],
       [-1.02082882, -1.21318999,  0.18741746],
       [ 1.49268741,  1.22782124,  1.58280405],
       [ 1.17682728,  0.00998415,  0.20071936],
       [ 0.70012304,  1.33163754, -0.31802204],
       [ 0.0088456 , -1.63345378,  0.04498207],
       [ 0.55902662,  0.46200391, -0.39949982],
       [ 1.47796614, -0.50457379, -1.64450612],
       [-1.61927178, -0.64363221, -0.11287487],
       [-1.267656  ,  1.13225036, -1.37920552],
       [-0.75856261, -0.74865064, -1.35676548],
       [-1.6477566 , -1.3853128 ,  0.27034854],
       [ 0.30559723,  1.1777216 , -1.65883577],
       [ 0.12137572,  0.84583242, -1.35676548],
       [-1.28790757, -1.43143631, -0.68257581],
       [-0.52094879, -1.29701657, -1.14201102],
       [ 0.41309098, -0.87549189, -1.31304986],
       [ 1.45584317,  1.1777216 ,  1.43335009],
       [ 0.96816082, -0.93708342, -1.08642518]])
```

Splitting Dataset

```python
In [205]:
X_train, X_test, y_train, y_test = train_test_split( X, y , test_size = 0.2, random_state = 42)
```

Training Model

Linear Regression

```python
In [206]:
lr = LinearRegression( fit_intercept = True)
lr.fit(X_train, y_train)
Out[206]:
LinearRegression()
```

```python
In [207]:
print(f"Linear coefficients : {lr.coef_}")
print(f"Intercept : {lr.intercept_}")
Linear coefficients : [3.8958379  2.76472707 0.13098562]
Intercept : 14.022794148616768
```

Model Prediction

```python
In [208]:
y_pred = lr.predict(X_test)
```

Metrices

RMSE

```python
In [209]:
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
Root Mean Squared Error: 1.6494590221354475
```

R-squared

```python
In [210]:
lrs=r2_score(y_test, y_pred)
print('R2 Score:',lrs*100)
R2 Score: 91.38021136866259
```

```python
In [211]:
lrscore = cross_val_score(lr, X, y, cv=5)
lrc = rfscore.mean()
print('Cross val score :', lrc*100 )
Cross val score : 97.68298993641935
```

```python
In [212]:
plt.figure(figsize=(8,6))
plt.scatter(x=y_test, y=y_pred,color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel('Actual Sale', fontsize=14)
plt.ylabel('Predicted Sale', fontsize=14)
plt.title('Linear Regression',fontsize=18)
plt.savefig('lr.png')
plt.show()
```

RandomForest Regressor

```python
In [213]:
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor

parameters = {'criterion':['mse', 'mae'],
              'max_features':["auto", "sqrt", "log2"]}
rf=RandomForestRegressor()
clf=GridSearchCV(rf,parameters)
clf.fit(X_train, y_train)
print(clf.best_params_)
{'criterion': 'mse', 'max_features': 'auto'}
```

```python
In [214]:
rf=RandomForestRegressor(criterion="mae", max_features="auto")
rf.fit(X_train, y_train)
rf.score(X_train, y_train)
y_pred = rf.predict(X_test)

rfs = r2_score(y_test, y_pred)
print('R2 Score:',lrs*100)

rfscore = cross_val_score(rf, X, y, cv=5)
rfc = rfscore.mean()
print('Cross val score :', rfc*100 )
R2 Score: 91.38021136866259
Cross val score : 97.6284579530873
```

```python
In [215]:
from sklearn.metrics import mean_squared_error
from sklearn.datasets import make_friedman1
from sklearn.ensemble import GradientBoostingRegressor

X, y = make_friedman1(n_samples=1200, random_state=0, noise=1.0)
X_train, X_test = X[:200], X[200:]
y_train, y_test = y[:200], y[200:]
est = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1,max_depth=1, random_state=0,
                                loss='squared_error').fit(X_train, y_train)
mean_squared_error(y_test, est.predict(X_test))
Out[215]:
5.009154859960321
```

Regularization

```python
In [216]:
parameters = {'alpha':[.0001, .001, .01, .1, 1, 10],
              'random_state':list(range(0,10))}
ls = Lasso()
clf = GridSearchCV(ls,parameters)
clf.fit(X_train,y_train)
print(clf.best_params_)
{'alpha': 0.01, 'random_state': 0}
```

```python
In [217]:
#model training
ls = Lasso(alpha=0.01,random_state=0)
ls.fit(X_train,y_train)
ls_score_training = ls.score(X_train,y_train)
pred_ls = ls.predict(X_test)
ls_score_training*100
Out[217]:
71.37804963414543
```

```python
In [219]:
cv_score=cross_val_score(ls,X,y,cv=5)
cv_mean=cv_score.mean()
cv_mean*100
Out[219]:
74.58420785115423
```

```python
In [229]:
import pickle
filename = 'sales.pkl'
pickle.dump(ls,open(filename, 'wb'))
```

```python
In [230]:
loaded_model = pickle.load(open('sales.pkl','rb'))
result = loaded_model.score(X_test,y_test)
print(result*100)
74.50360766719032
```

```python
In [231]:
import statsmodels.api as sm

X2 = sm.add_constant(X)
est = sm.OLS(y, X2)
est2 = est.fit()
print(est2.summary())
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.752
Model:                            OLS   Adj. R-squared:                  0.750
Method:                 Least Squares   F-statistic:                     360.3
Date:                Thu, 08 Sep 2022   Prob (F-statistic):               0.00
Time:                        23:20:17   Log-Likelihood:                -2828.8
No. Observations:                1200   AIC:                             5680.
Df Residuals:                    1189   BIC:                             5736.
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.6102      0.409      1.490      0.136      -0.193       1.414
x1             6.5365      0.258     25.365      0.000       6.031       7.042
x2             6.8189      0.258     26.389      0.000       6.312       7.326
x3            -0.1891      0.264     -0.716      0.474      -0.707       0.329
x4            10.4618      0.250     41.894      0.000       9.972      10.952
x5             4.7652      0.252     18.927      0.000       4.271       5.259
x6            -0.0808      0.261     -0.309      0.757      -0.594       0.432
x7            -0.1809      0.262     -0.690      0.491      -0.695       0.334
x8            -0.2793      0.256     -1.092      0.275      -0.781       0.223
x9            -0.1969      0.248     -0.795      0.427      -0.683       0.289
x10           -0.3499      0.264     -1.324      0.186      -0.869       0.169
==============================================================================
Omnibus:                       61.714   Durbin-Watson:                   1.982
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               88.590
Skew:                          -0.450   Prob(JB):                     5.79e-20
Kurtosis:                       3.980   Cond. No.                         12.2
==============================================================================

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

```python
In [ ]:
```