In [180…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as mlt
import seaborn as sn
%matplotlib inline
```

In [181…
```
# Loading dataset -
```

In [182…
```
advertising_df = pd.read_csv('/Users/arijeetbhadra/Downloads/advertising.
```

In [183…
```
advertising_df.head(10)
```

Out[183]:

	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
5	8.7	48.9	75.0	7.2
6	57.5	32.8	23.5	11.8
7	120.2	19.6	11.6	13.2
8	8.6	2.1	1.0	4.8
9	199.8	2.6	21.2	15.6

In [184…
```
advertising_df.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
```

In [185…
```
advertising_df.shape
```

Out[185]:
```
(200, 4)
```

In [186…
```
advertising_df.describe()
```

Out[186]:

	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

In [187…
```
advertising_df.isnull().sum()
```

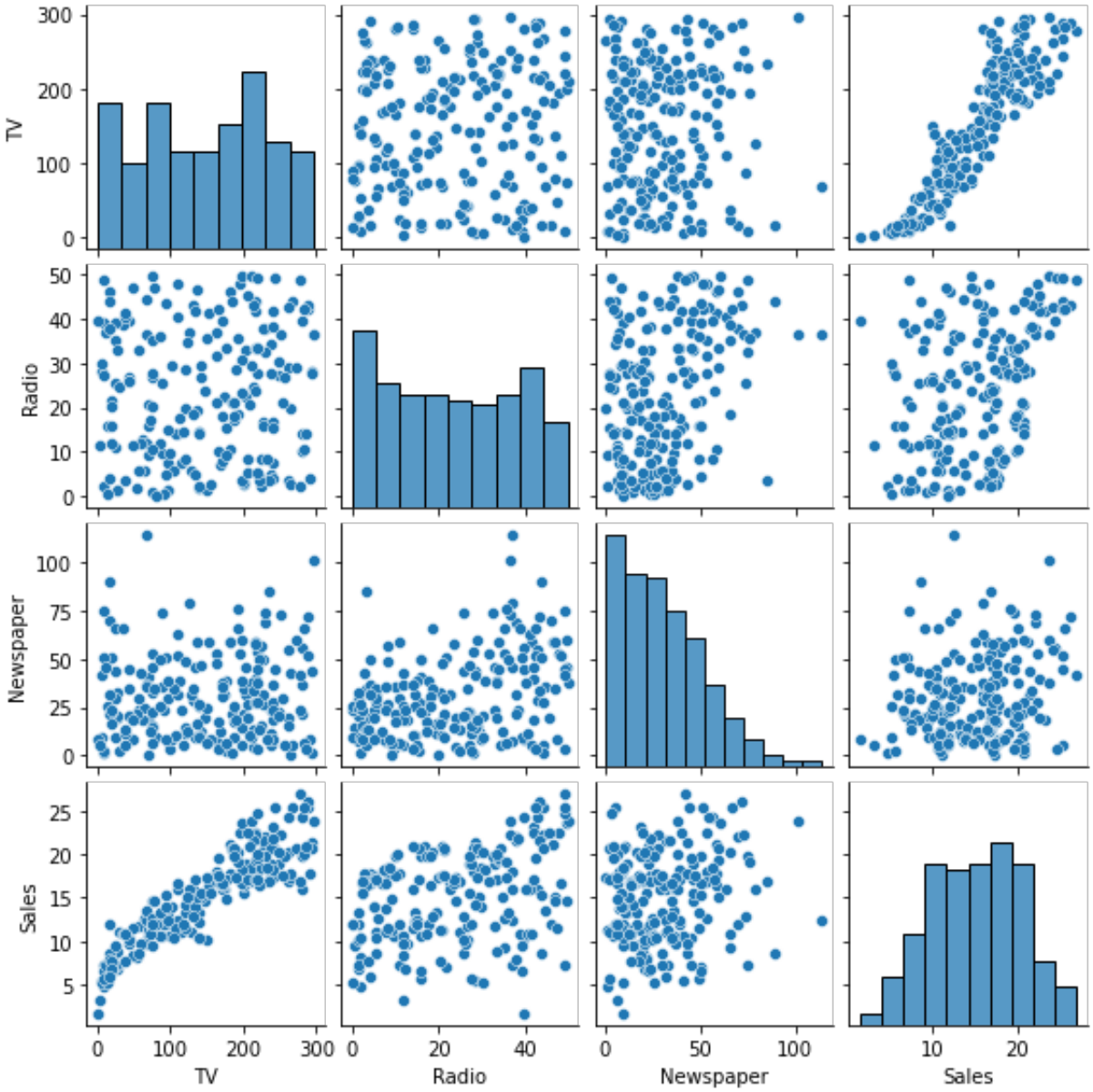
Out[187]:
```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

In [188…
```
# Finding co-relation -
```

In [189…
```
influencing_factors =['TV','Radio','Newspaper','Sales']
sn.pairplot(advertising_df[influencing_factors],size=2)
```

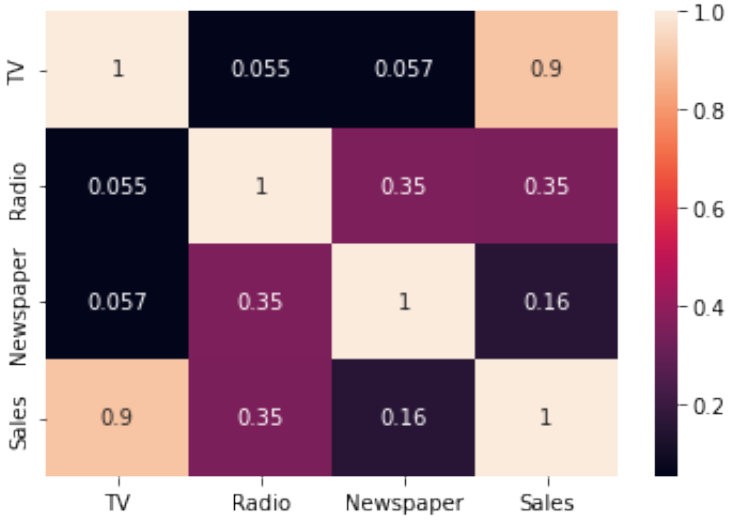
```
/opt/anaconda3/lib/python3.9/site-packages/seaborn/axisgrid.py:2076: UserWarning: The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

Out[189]:
```
<seaborn.axisgrid.PairGrid at 0x7ff26724f730>
```



In [190…
```
sn.heatmap(advertising_df[influencing_factors].corr(),annot=True)
```

Out[190]: <AxesSubplot:>



In [191…
```
# As observed TV shows high co-releation with sales, which is directing u
```

In [192…
```
advertising_df[['TV','Sales']].head(10).reset_index()
```

Out[192]:

	index	TV	Sales
0	0	230.1	22.1
1	1	44.5	10.4
2	2	17.2	12.0
3	3	151.5	16.5
4	4	180.8	17.9
5	5	8.7	7.2
6	6	57.5	11.8
7	7	120.2	13.2
8	8	8.6	4.8
9	9	199.8	15.6

In [193…
```
import statsmodels.api as sm
x=sm.add_constant(advertising_df['TV'])
x.head(10)
```

Out[193]:

	const	TV
0	1.0	230.1
1	1.0	44.5
2	1.0	17.2
3	1.0	151.5
4	1.0	180.8
5	1.0	8.7
6	1.0	57.5
7	1.0	120.2
8	1.0	8.6
9	1.0	199.8

In [194…
```
y=advertising_df['Sales']
y.head(10).reset_index()
```

Out[194]:

	index	Sales
0	0	22.1
1	1	10.4
2	2	12.0
3	3	16.5
4	4	17.9
5	5	7.2
6	6	11.8
7	7	13.2
8	8	4.8
9	9	15.6

In [195…
```
# Split train & test data -
from sklearn.model_selection import train_test_split
```

In [196…
```
train_x,test_x,train_y,test_y= train_test_split(x,y,train_size=0.8,random
```

In [197…
```
# Fitting the model
advertising_df_1m= sm.OLS(train_y, train_x).fit()
```

In [198…
```
print(advertising_df_1m.params)
```

```
const    6.995533
TV       0.054105
dtype: float64
```

In [199…
```
# Model Prediction -
# SALES = 6.9955 + 0.054*(TV)
```

In [200…
```
# Model diagnostics -
advertising_df_1m.summary2()
```

Out[200]:

Model:	OLS	Adj. R-squared:	0.820
Dependent Variable:	Sales	AIC:	723.1364
Date:	2024-09-23 03:15	BIC:	729.2867
No. Observations:	160	Log-Likelihood:	-359.57
Df Model:	1	F-statistic:	727.7
Df Residuals:	158	Prob (F-statistic):	5.03e-61
R-squared:	0.822	Scale:	5.3085

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
const	6.9955	0.3431	20.3863	0.0000	6.3178	7.6733
TV	0.0541	0.0020	26.9763	0.0000	0.0501	0.0581

Omnibus:	0.122	Durbin-Watson:	2.203
Prob(Omnibus):	0.941	Jarque-Bera (JB):	0.029
Skew:	0.032	Prob(JB):	0.986
Kurtosis:	3.014	Condition No.:	322

In [201…
```
# Model diagnostics
# R- Square - coefficient of determination - 82.2%, which is decent value
```

In [202…
```
# p values is 0.0581 which indicates that there is statistically signific
```

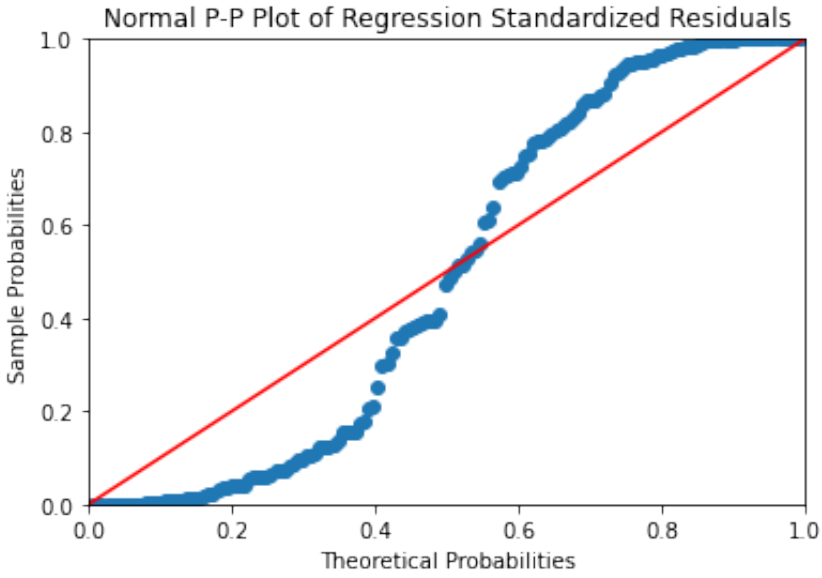
In [203…
```
# For SLR - p value for t-test and F-test will be same since null hypothe
```

In [204…
```
# Check For normal distribution of residual-
```

In [205…
```
import matplotlib.pyplot as plt
import seaborn as sn
%matplotlib inline
```

In [206…
```
advertising_df_resid = advertising_df_1m.resid
probplot = sm.ProbPlot(advertising_df_resid)
plt.figure(figsize = (8,6))
probplot.ppplot(line='45')
plt.title('Normal P-P Plot of Regression Standardized Residuals')
plt.show()
```

```
<Figure size 576x432 with 0 Axes>
```



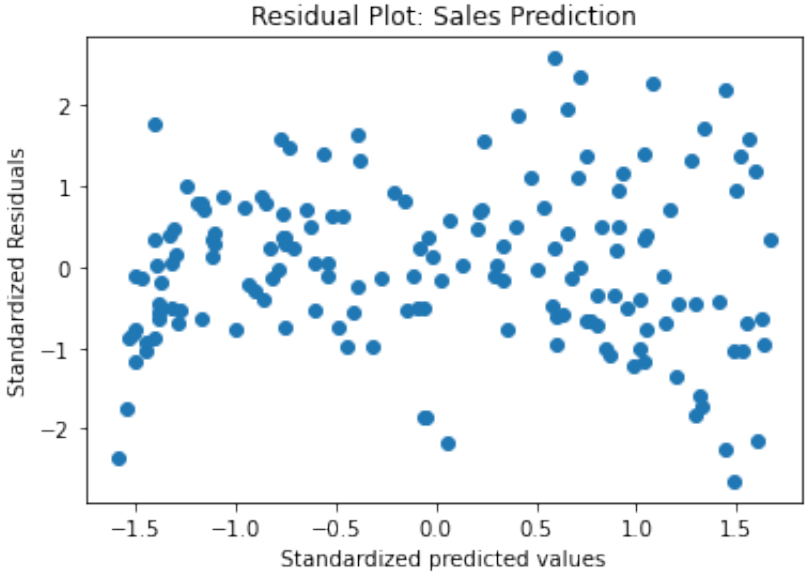
In [207…
```
# Diagnoal line is the cummulative distribution of normal distribution,
# whereas dots represents cummulative distribution of residuals.
# Since dots are close to line, we can say that residuals are normally di
```

In [208…
```
# Test of Homoscedasticity
```

In [209…
```
def get_standardized_values(vals):
    return(vals-vals.mean())/vals.std()
```

In [210…
```
plt.scatter(get_standardized_values(advertising_df_1m.fittedvalues),get_s
plt.title('Residual Plot: Sales Prediction')
plt.xlabel('Standardized predicted values')
plt.ylabel('Standardized Residuals')
```

Out[210]:
```
Text(0, 0.5, 'Standardized Residuals')
```



In [211…
```
# Shape is not simillar to funnel type structure, which validates homosce
```

In [212…
```
# outlier analysis through Z test
```

In [213…
```
from scipy.stats import zscore
advertising_df['z_score_Sales']=zscore(advertising_df.Sales)
advertising_df[(advertising_df.z_score_Sales>3.0)|(advertising_df.z_score
```

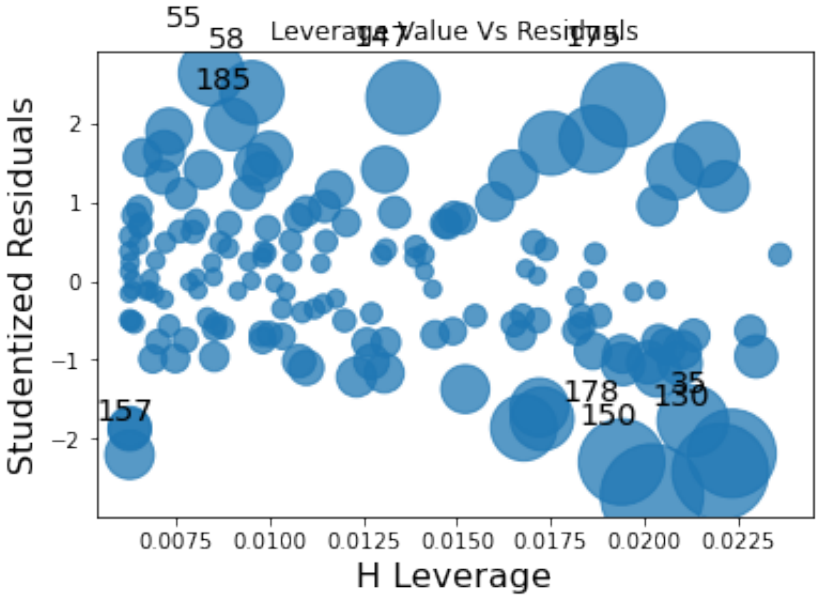
Out[213]:

	TV	Radio	Newspaper	Sales	z_score_Sales

In [214…
```
# Hence no outliers detected
```

In [215…
```
# Leverage Values
```

In [216…
```
from statsmodels.graphics.regressionplots import influence_plot
fig, ax = plt.subplots( figsize=(6,4) )
influence_plot(advertising_df_1m,ax = ax )
plt.title('Leverage Value Vs Residuals')
plt.show();
```



In [229…
```
# Making prediction using the model
```

In [233…
```
from sklearn.metrics import r2_score,mean_squared_error
pred_y=advertising_df_1m.predict(test_x)
```

In [234…
```
# Finding R-Square and RMSE
```

In [235…
```
np.abs(r2_score(test_y,pred_y))
```

Out[235]:
```
0.7281352744078886
```

In [226…
```
# So the model can explains 72.8% of variance in the validation set
```

In [239…
```
np.sqrt(mean_squared_error(test_y,pred_y))
```

Out[239]: 2.3126831803046097

In [240…
```
# RSME means average error of the model makes in predicting the outcome.
```

In [ ]:
```
# Calculating prediction intervals
```

In [248…
```
from statsmodels.sandbox.regression.predstd import wls_prediction_std
# Predict the y values
pred_y = advertising_df_1m.predict(test_x)

# Predict the low and high interval values for y
_, pred_y_low, pred_y_high = wls_prediction_std(advertising_df_1m,test_x,

# Store all the values in a dataframe
pred_y_df = pd.DataFrame( { 'Marketing through TV': test_x['TV'],
                            'pred_y': pred_y,
                            'pred_y_left': pred_y_low,
                            'pred_y_right': pred_y_high } )
```

In [249…
```
pred_y_df[0:10]
```

Out[249]:

	Marketing through TV	pred_y	pred_y_left	pred_y_right
126	7.8	7.417556	3.566539	11.268572
104	238.2	19.883459	16.046962	23.719956
99	135.2	14.310594	10.486447	18.134741
92	217.7	18.774296	14.942684	22.605909
111	241.7	20.072828	16.235377	23.910279
167	206.8	18.184547	14.355042	22.014052
116	139.2	14.527016	10.702959	18.351073
96	197.6	17.686776	13.858785	21.514768
52	216.4	18.703959	14.872616	22.535303
69	216.8	18.725602	14.894176	22.557027

In [ ]:
```

```