SimpleLinearRegression

October 1, 2024

[2]:
```python
# Different types of data sets available in Seaborn library

import seaborn as sns
sns.get_dataset_names()
```

[2]:
```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'dowjones',
 'exercise',
 'flights',
 'fmri',
 'geyser',
 'glue',
 'healthexp',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'seaice',
 'taxis',
 'tips',
 'titanic',
 'anagrams',
 'anagrams',
 'anscombe',
 'anscombe',
 'attention',
 'attention',
 'brain_networks',
 'brain_networks',
 'car_crashes',
 'car_crashes',
```

```
'diamonds',
'diamonds',
'dots',
'dots',
'dowjones',
'dowjones',
'exercise',
'exercise',
'flights',
'flights',
'fmri',
'fmri',
'geyser',
'geyser',
'glue',
'glue',
'healthexp',
'healthexp',
'iris',
'iris',
'mpg',
'mpg',
'penguins',
'penguins',
'planets',
'planets',
'seaice',
'seaice',
'taxis',
'taxis',
'tips',
'tips',
'titanic',
'titanic',
'anagrams',
'anscombe',
'attention',
'brain_networks',
'car_crashes',
'diamonds',
'dots',
'dowjones',
'exercise',
'flights',
'fmri',
'geyser',
'glue',
```

```
 'healthexp',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'seaice',
 'taxis',
 'tips',
 'titanic']
```

[3]:
```
tips = sns.load_dataset('tips')
tips.head()
```

[3]:
```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female     No  Sun  Dinner     4
```

[4]:
```
taxis = sns.load_dataset('taxis')
taxis.head()
```

[4]:
```
               pickup             dropoff  passengers  distance  fare   tip  \
0 2019-03-23 20:21:09 2019-03-23 20:27:24           1      1.60   7.0  2.15
1 2019-03-04 16:11:55 2019-03-04 16:19:00           1      0.79   5.0  0.00
2 2019-03-27 17:53:01 2019-03-27 18:00:25           1      1.37   7.5  2.36
3 2019-03-10 01:23:59 2019-03-10 01:49:51           1      7.70  27.0  6.15
4 2019-03-30 13:27:42 2019-03-30 13:37:14           3      2.16   9.0  1.10

   tolls  total   color      payment            pickup_zone  \
0    0.0  12.95  yellow  credit card        Lenox Hill West
1    0.0   9.30  yellow         cash  Upper West Side South
2    0.0  14.16  yellow  credit card          Alphabet City
3    0.0  36.95  yellow  credit card              Hudson Sq
4    0.0  13.40  yellow  credit card           Midtown East

            dropoff_zone pickup_borough dropoff_borough
0    UN/Turtle Bay South      Manhattan       Manhattan
1  Upper West Side South      Manhattan       Manhattan
2           West Village      Manhattan       Manhattan
3         Yorkville West      Manhattan       Manhattan
4         Yorkville West      Manhattan       Manhattan
```

[5]:
```
a = sns.load_dataset('iris')
a.head()
```

[5]:
```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
```

[6]:
```python
# Simple Linear Regression Model
# Y = mX + c This is the straight line formula
#Y is the response variable
# X is the predictor , here its TV column
# C is the intercept
# m is the co-effiecient for X

# Y(Sales) = C + m * TV
```

[7]:
```python
import numpy as np, pandas as pd
import warnings
warnings.filterwarnings('ignore')
advertising = pd.read_csv("advertising.csv")
advertising.head()
```

[7]:
```
      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
```

[8]:
```python
advertising.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
```

[9]:
```python
advertising.isnull().sum()
```

[9]:
```
TV           0
Radio        0
```

```
Newspaper    0
Sales        0
dtype: int64
```

[10]:
```
advertising.shape
```

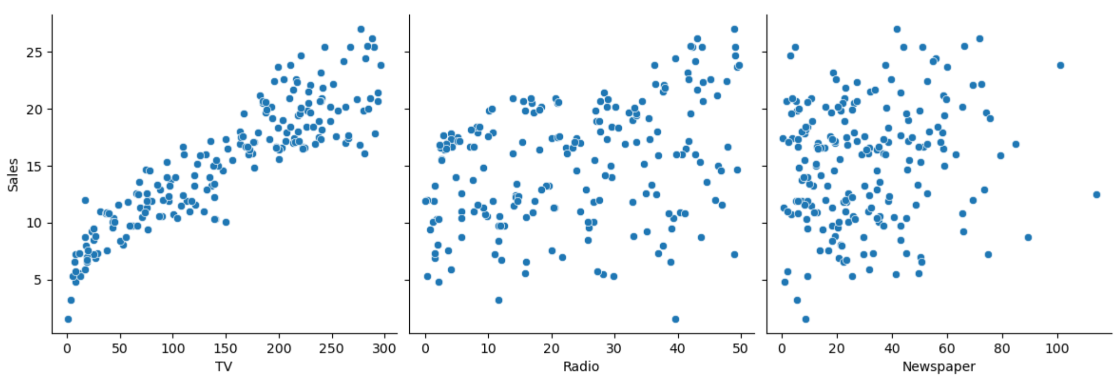
[10]:
```
(200, 4)
```

[11]:
```
advertising.describe()
```

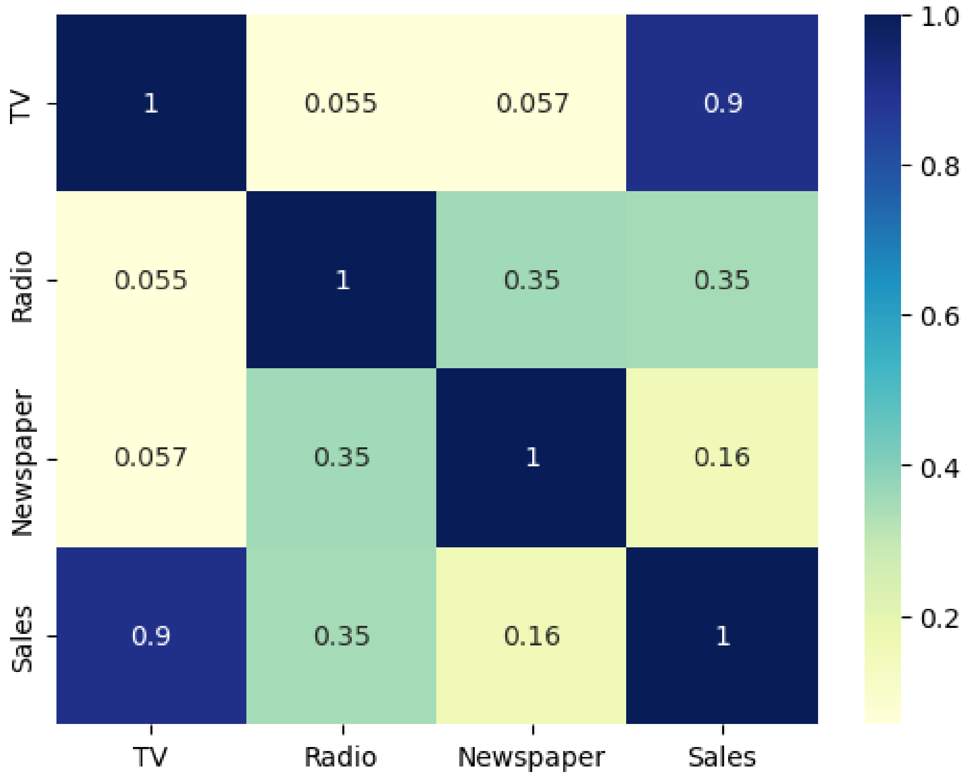
[11]:

	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

[12]:
```
import matplotlib.pyplot as plt, seaborn as sns
sns.pairplot(advertising, x_vars = ['TV','Radio','Newspaper'], y_vars =␣
↪['Sales'],kind = 'scatter',size = 4 )
plt.show()
```



[13]:
```
sns.heatmap(advertising.corr(),cmap = "YlGnBu", annot = True)
plt.show()
```



[41]:
```
X = advertising['TV']
y = advertising['Sales']
```

[43]:
```
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
```

[16]:
```
#pip install scikit-learn
```

[17]:
```
#pip install statsmodels
```

[57]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size =0.2,␣
↪train_size = 0.8, random_state= 42)
```

[59]:
```
X_train.head()
```

[59]:
```
79     116.0
197    177.0
38      43.1
24      62.3
122    224.0
Name: TV, dtype: float64
```

[61]:
```
y_train.head()
```

[61]:
```
79     11.0
197    14.8
38     10.1
24      9.7
122    16.6
Name: Sales, dtype: float64
```

[63]:
```
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

[63]:
```
((160,), (160,), (40,), (40,))
```

[71]:
```
X_train_sm = sm.add_constant(X_train)
X_train_sm # C term is added , where it touches the y-axis at 1
```

[71]:
```
     const     TV
79     1.0  116.0
197    1.0  177.0
38     1.0   43.1
24     1.0   62.3
122    1.0  224.0
..     …     …
106    1.0   25.0
14     1.0  204.1
92     1.0  217.7
179    1.0  165.6
102    1.0  280.2

[160 rows x 2 columns]
```

[73]:
```
# For linear reg you will Ordinary Least Squares Method

lr = sm.OLS(y_train, X_train_sm).fit()
lr.params # you have got the m's or the co-effcients
```

[73]:
```
const    7.007108
TV       0.055483
dtype: float64
```

[75]:
```
print(lr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.813
Model:                            OLS   Adj. R-squared:                  0.812
Method:                 Least Squares   F-statistic:                     689.1
Date:                Tue, 01 Oct 2024   Prob (F-statistic):           1.71e-59
```

```
Time:                        03:44:39   Log-Likelihood:                -355.76
No. Observations:                 160   AIC:                             715.5
Df Residuals:                     158   BIC:                             721.7
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          7.0071      0.364     19.274      0.000       6.289       7.725
TV             0.0555      0.002     26.251      0.000       0.051       0.060
==============================================================================
Omnibus:                        0.631   Durbin-Watson:                   2.262
Prob(Omnibus):                  0.730   Jarque-Bera (JB):                0.767
Skew:                          -0.110   Prob(JB):                        0.681
Kurtosis:                       2.742   Cond. No.                         352.
==============================================================================

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

Looking at some key statistics from the summary The values we are concerned with are - 1. The coefficients and significance (p-values) 2. R-squared 3. F statistic and its significance

1. The coefficient for TV is 0.0555, with a very low p value The coefficient is statistically significant. So the association is not purely by chance.

2. R - squared is 0.813 Meaning that 81.3% of the variance in `Sales` is explained by `TV`

This is a decent R-squared value.

3. F statistic has a very low p value (practically low)

Meaning that the model fit is statistically significant, and the explained variance isn't purely by chance.

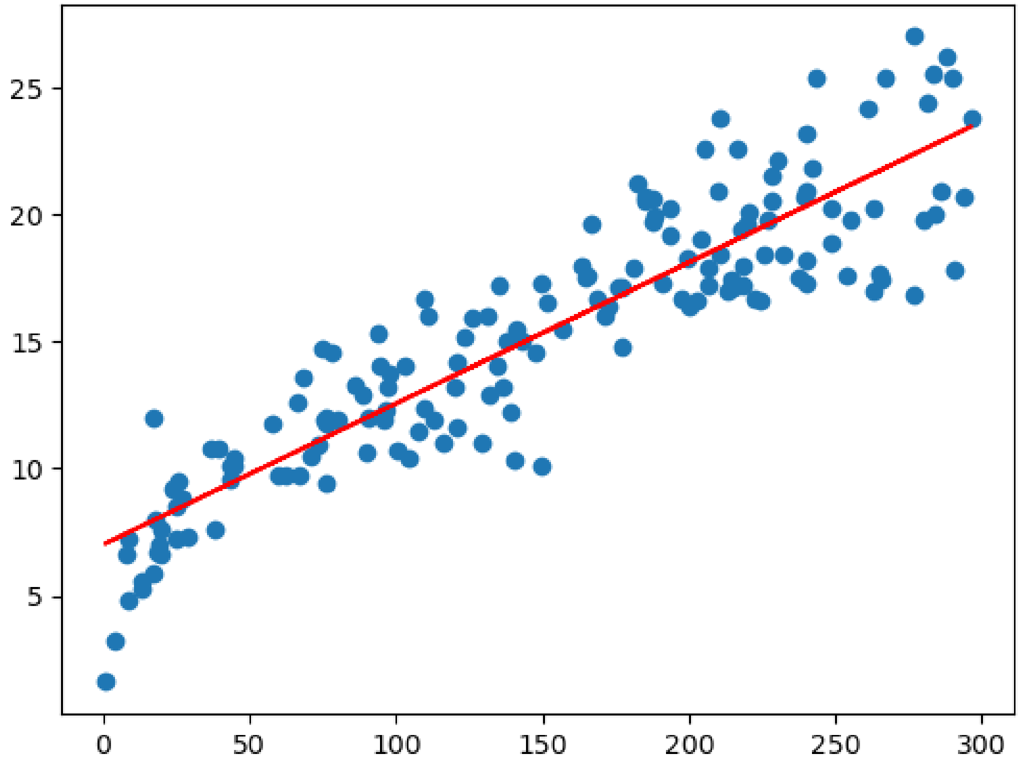
The fit is significant. Let's visualize how well the model fit the data.

From the parameters that we get, our linear regression equation becomes:

Sales = 7.0071 + 0.0555 * TV

[77]:
```python
# Sales = 7.0071 + 0.0555 * TV

plt.scatter(X_train, y_train)
plt.plot(X_train,7.0071 + 0.0555 * X_train,'r')
plt.show()
```



25
20
15
10
5
0
50
100
150
200
250
300


```
[ ]:
```