SimpleLinearRegression (1)

August 19, 2022

[1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```python
data = pd.read_csv('Advertising.csv',index_col=0)
data.head()
```

[2]:

	TV	radio	newspaper	sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

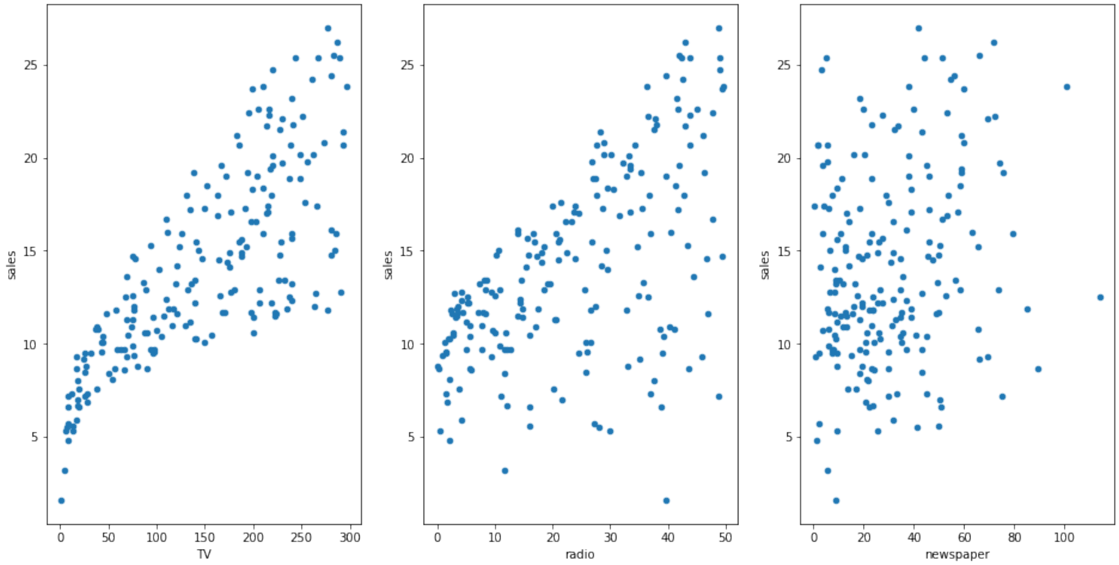
[3]:
```python
data.columns
```

[3]:
```
Index(['TV', 'radio', 'newspaper', 'sales'], dtype='object')
```

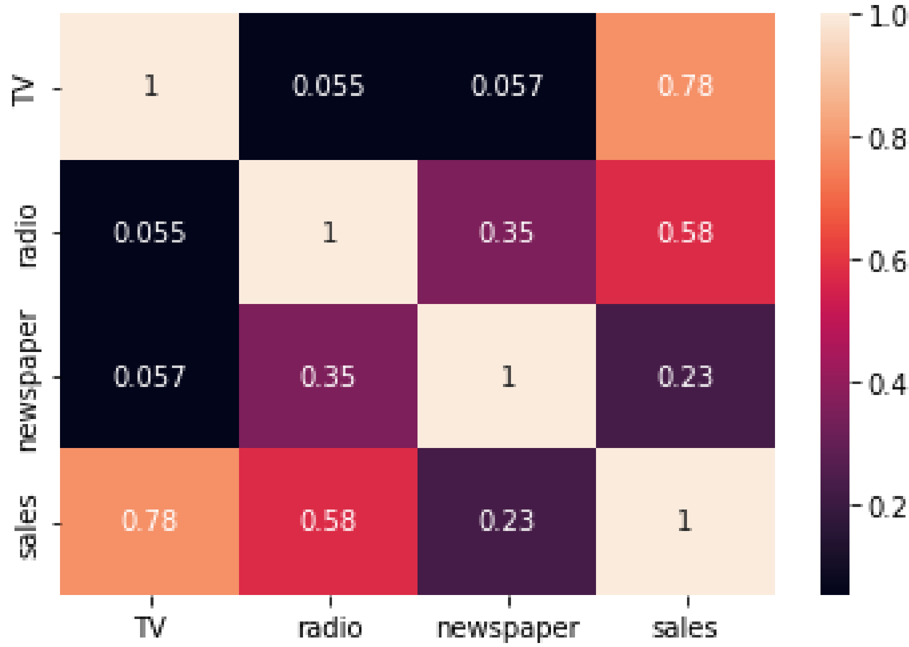
[4]:
```python
data.shape
```

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

[5]:
```python
fig, axs = plt.subplots(1, 3)
data.plot(kind='scatter', x='TV', y='sales', ax=axs[0], figsize=(16,8))
data.plot(kind='scatter', x='radio', y='sales', ax=axs[1])
data.plot(kind='scatter', x='newspaper', y='sales', ax=axs[2]);
```



[6]:
```
sns.heatmap(data.corr(), annot = True);
```



[7]:
```
features = data[['radio']].values
target = data[['sales']].values
```

[8]:
```
from sklearn.model_selection import train_test_split
```

[9]:
```
X_train, X_test, y_train, y_test = train_test_split(features, target,␣
↪random_state = 6)
```

[10]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(150, 1)
(50, 1)
(150, 1)
(50, 1)
```

[11]:
```
# X_train = X_train.reshape(150,-1)
# X_train.shape
```

[12]:
```
# X_test = X_test.reshape(50,-1)
# X_test.shape
```

[13]:
```
from sklearn.linear_model import LinearRegression
```

[14]:
```
my_first_model = LinearRegression()
```

At this stage my m and c values are initialized to some random value. We need to train the model to find the optimal value of the weights(parameters) of the Linear Regression model.

[15]:
```
my_first_model.fit(X_train,y_train)
```

[15]:
```
LinearRegression()
```

[16]:
```
my_first_model.coef_
```

[16]:
```
array([[0.21590796]])
```

[17]:
```
my_first_model.intercept_
```

[17]:
```
array([9.17863927])
```

Interpreting the coefficients

1. A unit increase in TV ad spending was associated with a .048 unit increase in Sales

OR

An additional 1000 $ spent on TV was associated with an increase in sales of 48.734 units

y = mx + c

You have the values of m and c. Given any value of x you can predict the value of y

In a new market my spend on TV is $50,000. I want you to tell me the sales generated due to this spend

[18]:
```
.048*50 + 6.709
```

[18]:
```
9.109
```

Lets try to plot best fit line

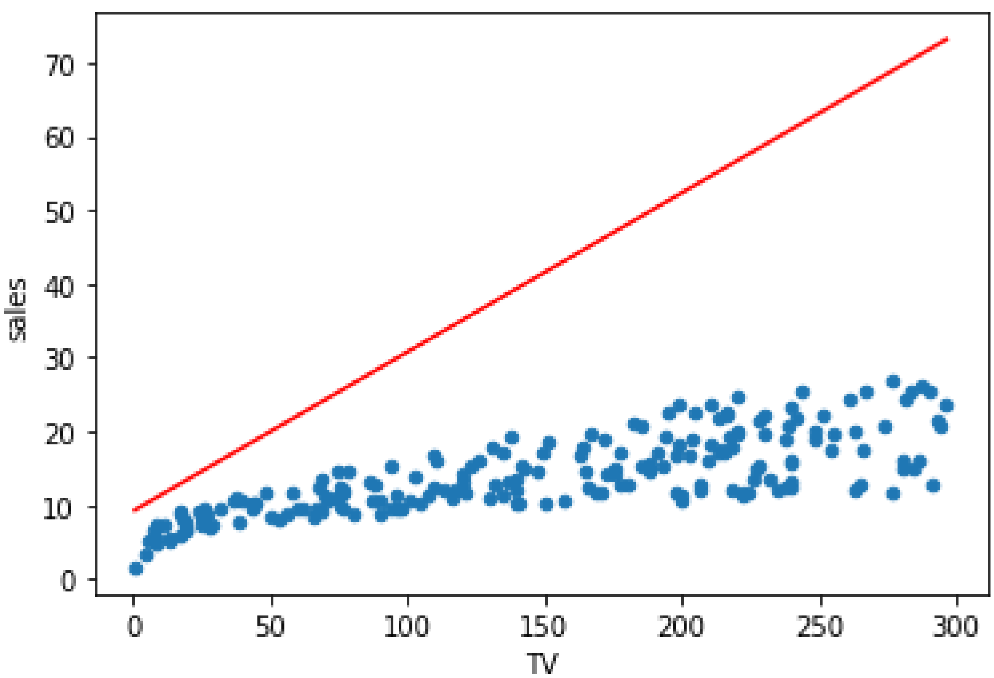
[19]:
```
X_new = pd.DataFrame({'TV':[data.TV.min(),data.TV.max()]})
X_new.head()
```

[19]:

	TV
0	0.7
1	296.4

[20]:
```
preds = my_first_model.predict(X_new)
```

[21]:
```
data.plot(kind='scatter', x='TV', y='sales')
plt.plot(X_new,preds,c='red');
```



[22]:
```
import statsmodels.formula.api as smf
```

[23]:
```
my_stat_model = smf.ols(formula='sales ~ TV', data=data).fit()
my_stat_model.pvalues
```

[23]:
```
Intercept    1.406300e-35
TV           1.467390e-42
dtype: float64
```

H0 : There is no relationship between independent(TV) variable and dependent(sales) variable

H1 : There is a relationship between independent(TV) variable and dependent(Sales) variable

if my p value for TV is less than .05 then I will reject the null hypothesis

[24]:
```python
my_stat_model.summary()
```

[24]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  sales   R-squared:                       0.612
Model:                            OLS   Adj. R-squared:                  0.610
Method:                 Least Squares   F-statistic:                     312.1
Date:                Tue, 05 Apr 2022   Prob (F-statistic):           1.47e-42
Time:                        01:19:56   Log-Likelihood:                -519.05
No. Observations:                 200   AIC:                             1042.
Df Residuals:                     198   BIC:                             1049.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      7.0326      0.458     15.360      0.000       6.130       7.935
TV             0.0475      0.003     17.668      0.000       0.042       0.053
==============================================================================
Omnibus:                        0.531   Durbin-Watson:                   1.935
Prob(Omnibus):                  0.767   Jarque-Bera (JB):                0.669
Skew:                          -0.089   Prob(JB):                        0.716
Kurtosis:                       2.779   Cond. No.                         338.
==============================================================================

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

To complete the supervised learning flow follow the below steps after fitting the model

[25]:
```python
my_model_preds = my_first_model.predict(X_test)
```

[26]:
```python
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

[27]:
```
#MAE
mean_absolute_error(my_model_preds,y_test)
```

[27]: 3.2941636885230885

[28]:
```
#MSE
mean_squared_error(my_model_preds,y_test)
```

[28]: 17.228484653665376

[29]:
```
#RMSE
np.sqrt(mean_squared_error(my_model_preds,y_test))
```

[29]: 4.150720979982318

SLR:TV

[30]:
```
feature1=data[['TV']].values
target1=data[['sales']].values
from sklearn.model_selection import train_test_split
x_train1,x_test1,y_train1,y_test1=train_test_split(feature1,target1,random_state=6)
from sklearn.linear_model import LinearRegression
lr_TV=LinearRegression()
#fitting the curve
lr_TV.fit(x_train1,y_train1)
print('Coefficient_TV:',lr_TV.coef_)
print('Intercept_TV:',lr_TV.intercept_)
#predicting values
y_pred1=lr_TV.predict(x_test1)
from sklearn.metrics import mean_absolute_error,mean_squared_error
print('MAE:',mean_absolute_error(y_test1,y_pred1))
print('MSE:',mean_squared_error(y_test1,y_pred1))
print('RMSE:',np.sqrt(mean_squared_error(y_test1,y_pred1)))
my_stat_model1 = smf.ols(formula='sales ~ TV', data=data).fit()
my_stat_model1.pvalues
print(my_stat_model1.summary())
```

```
Coefficient_TV: [[0.04873499]]
Intercept_TV: [6.70910349]
MAE: 2.469197684055691
MSE: 9.50319169686634
RMSE: 3.082724719605424
                            OLS Regression Results
==============================================================================
Dep. Variable:                  sales   R-squared:                       0.612
Model:                            OLS   Adj. R-squared:                  0.610
Method:                 Least Squares   F-statistic:                     312.1
Date:                Tue, 05 Apr 2022   Prob (F-statistic):           1.47e-42
```

```
Time:                        01:19:56   Log-Likelihood:                -519.05
No. Observations:                 200   AIC:                             1042.
Df Residuals:                     198   BIC:                             1049.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      7.0326      0.458     15.360      0.000       6.130       7.935
TV             0.0475      0.003     17.668      0.000       0.042       0.053
==============================================================================
Omnibus:                        0.531   Durbin-Watson:                   1.935
Prob(Omnibus):                  0.767   Jarque-Bera (JB):                0.669
Skew:                          -0.089   Prob(JB):                        0.716
Kurtosis:                       2.779   Cond. No.                         338.
==============================================================================

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

newspaper

[31]:
```python
feature2=data[['newspaper']].values
target2=data[['sales']].values
from sklearn.model_selection import train_test_split
x_train2,x_test2,y_train2,y_test2=train_test_split(feature2,target2,random_state=6)
from sklearn.linear_model import LinearRegression
lr_NP=LinearRegression()
#fitting the curve
lr_NP.fit(x_train2,y_train2)
print('Coefficient_NP:',lr_NP.coef_)
print('Intercept_NP:',lr_NP.intercept_)
#predicting values
y_pred2=lr_NP.predict(x_test2)
from sklearn.metrics import mean_absolute_error,mean_squared_error
print('MAE:',mean_absolute_error(y_test2,y_pred2))
print('MSE:',mean_squared_error(y_test2,y_pred2))
print('RMSE:',np.sqrt(mean_squared_error(y_test2,y_pred2)))
my_stat_model2 = smf.ols(formula='sales ~ newspaper', data=data).fit()
my_stat_model2.pvalues
print(my_stat_model2.summary())
```

```
Coefficient_NP: [[0.07750161]]
Intercept_NP: [11.88037471]
MAE: 4.156100638515478
MSE: 24.68120739839393
RMSE: 4.968018457936115
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  sales   R-squared:                       0.052
Model:                            OLS   Adj. R-squared:                  0.047
Method:                 Least Squares   F-statistic:                     10.89
Date:                Tue, 05 Apr 2022   Prob (F-statistic):            0.00115
Time:                        01:19:56   Log-Likelihood:                -608.34
No. Observations:                 200   AIC:                             1221.
Df Residuals:                     198   BIC:                             1227.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     12.3514      0.621     19.876      0.000      11.126      13.577
newspaper      0.0547      0.017      3.300      0.001       0.022       0.087
==============================================================================
Omnibus:                        6.231   Durbin-Watson:                   1.983
Prob(Omnibus):                  0.044   Jarque-Bera (JB):                5.483
Skew:                           0.330   Prob(JB):                       0.0645
Kurtosis:                       2.527   Cond. No.                         64.7
==============================================================================

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

Multiple Linear Regression

[ ]:
```

```