Assignment 01

October 16, 2020

1 Assignment 01: Evaluate the Ad Budget Dataset of XYZ Firm

The comments/sections provided are your cues to perform the assignment. You don't need to limit yourself to the number of rows/cells provided. You can add additional rows in each section to add more lines of code.

If at any point in time you need help on solving this assignment, view our demo video to understand the different steps of the code.

Happy coding!

1: Import the dataset

[14]:
```
#Import the required libraries
import pandas as pd
```

[16]:
```
#Import the advertising dataset
df_ad_data = pd.read_csv("AdvertisingBudgetAndSales.csv", index_col=0)
```

2: Analyze the dataset

[17]:
```
#View the initial few records of the dataset
df_ad_data.head()
```

[17]:

	TV Ad Budget ($)	Radio Ad Budget ($)	Newspaper Ad Budget ($)	Sales ($)
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

[24]:
```
#Check the total number of elements in the dataset
df_ad_data.count()
```

[24]:
```
TV Ad Budget ($)           200
Radio Ad Budget ($)        200
Newspaper Ad Budget ($)    200
```

```
Sales ($)                  200
dtype: int64
```

3: Find the features or media channels used by the firm

[19]:
```
#Check the number of observations (rows) and attributes (columns) in the dataset
df_ad_data.size
```

[19]:
```
800
```

[20]:
```
df_ad_data.shape
```

[20]:
```
(200, 4)
```

[21]:
```
#View the names of each of the attributes
df_ad_data.columns
```

[21]:
```
Index(['TV Ad Budget ($)', 'Radio Ad Budget ($)', 'Newspaper Ad Budget ($)',
       'Sales ($)'],
      dtype='object')
```

[22]:
```
df_ad_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 200 entries, 1 to 200
Data columns (total 4 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   TV Ad Budget ($)         200 non-null    float64
 1   Radio Ad Budget ($)      200 non-null    float64
 2   Newspaper Ad Budget ($)  200 non-null    float64
 3   Sales ($)                200 non-null    float64
dtypes: float64(4)
memory usage: 7.8 KB
```

4: Create objects to train and test the model; find the sales figures for each channel

[57]:
```
#Create a feature object from the columns
X_feature = pd.DataFrame({
    "Newspaper":df_ad_data["Newspaper Ad Budget ($)"],
    "Radio":df_ad_data["Radio Ad Budget ($)"],
    "TV":df_ad_data["TV Ad Budget ($)"]
})
```

[58]:
```
#View the feature object
X_feature.head()
```

[58]:

	Newspaper	Radio	TV
1	69.2	37.8	230.1
2	45.1	39.3	44.5
3	69.3	45.9	17.2
4	58.5	41.3	151.5
5	58.4	10.8	180.8

[61]:
```
#View the target object
#from sklearn.datasets import load_iris
#iris_dataset = load_iris()
#iris_dataset.feature_names
#Y_target = iris_dataset.target
Y_target = df_ad_data[["Sales ($)"]]
Y_target.head()
```

[61]:

	Sales ($)
1	22.1
2	10.4
3	9.3
4	18.5
5	12.9

[62]:
```
#Verify if all the observations have been captured in the feature object
X_feature.shape
```

[62]: (200, 3)

[63]:
```
#Verify if all the observations have been captured in the target object
Y_target.shape
```

[63]: (200, 1)

5: Split the original dataset into training and testing datasets for the model

[64]:
```
#Split the dataset (by default, 75% is the training data and 25% is the testing␣
 ↪data)
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test =␣
 ↪train_test_split(X_feature,Y_target,random_state=1)
```

[82]:
```
#Verify if the training and testing datasets are split correctly (Hint: use the␣
 ↪shape() method)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(150, 3)
(50, 3)
(150, 1)
(50, 1)
```

6: Create a model to predict the sales outcome

[70]:
```
#Create a linear regression model
from sklearn.linear_model import LinearRegression
```

[71]:
```
linreg = LinearRegression()
linreg.fit(x_train,y_train)
```

[71]:
```
LinearRegression()
```

[72]:
```
#Print the intercept and coefficients
print(linreg.intercept_)
print(linreg.coef_)
```

```
[2.87696662]
[[0.00345046 0.17915812 0.04656457]]
```

[73]:
```
#Predict the outcome for the testing dataset
y_pred = linreg.predict(x_test)
```

[74]:
```
y_pred
```

[74]:
```
array([[21.70910292],
       [16.41055243],
       [ 7.60955058],
       [17.80769552],
       [18.6146359 ],
       [23.83573998],
       [16.32488681],
       [13.43225536],
       [ 9.17173403],
       [17.333853  ],
       [14.44479482],
       [ 9.83511973],
       [17.18797614],
       [16.73086831],
       [15.05529391],
       [15.61434433],
       [12.42541574],
       [17.17716376],
       [11.08827566],
       [18.00537501],
       [ 9.28438889],
```

```
       [12.98458458],
       [ 8.79950614],
       [10.42382499],
       [11.3846456 ],
       [14.98082512],
       [ 9.78853268],
       [19.39643187],
       [18.18099936],
       [17.12807566],
       [21.54670213],
       [14.69809481],
       [16.24641438],
       [12.32114579],
       [19.92422501],
       [15.32498602],
       [13.88726522],
       [10.03162255],
       [20.93105915],
       [ 7.44936831],
       [ 3.64695761],
       [ 7.22020178],
       [ 5.9962782 ],
       [18.43381853],
       [ 8.39408045],
       [14.08371047],
       [15.02195699],
       [20.35836418],
       [20.57036347],
       [19.60636679]])
```

7: Calculate the Mean Square Error (MSE)

[75]:
```
#Import required libraries for calculating MSE (mean square error)
from sklearn import metrics
import numpy as np
```

[76]:
```
#Calculate the MSE
print(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
1.404651423032897
```

[97]:
```
y_pred
```

[97]:
```
array([[21.70910292],
       [16.41055243],
       [ 7.60955058],
       [17.80769552],
```

```
[18.6146359 ],
[23.83573998],
[16.32488681],
[13.43225536],
[ 9.17173403],
[17.333853  ],
[14.44479482],
[ 9.83511973],
[17.18797614],
[16.73086831],
[15.05529391],
[15.61434433],
[12.42541574],
[17.17716376],
[11.08827566],
[18.00537501],
[ 9.28438889],
[12.98458458],
[ 8.79950614],
[10.42382499],
[11.3846456 ],
[14.98082512],
[ 9.78853268],
[19.39643187],
[18.18099936],
[17.12807566],
[21.54670213],
[14.69809481],
[16.24641438],
[12.32114579],
[19.92422501],
[15.32498602],
[13.88726522],
[10.03162255],
[20.93105915],
[ 7.44936831],
[ 3.64695761],
[ 7.22020178],
[ 5.9962782 ],
[18.43381853],
[ 8.39408045],
[14.08371047],
[15.02195699],
[20.35836418],
[20.57036347],
[19.60636679]])
```

[ ]: