Assignment_08_01_Completed

July 19, 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

[1]:
```
#Import the required libraries
import pandas as pd
```

[16]:
```
#Import the advertising dataset
df_ad_budget_sales = pd.read_csv("Advertising Budget and Sales.csv",index_col=0)
```

2: Analyze the dataset

[17]:
```
#View the initial few records of the dataset
df_ad_budget_sales.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
```

[18]:
```
#Check the total number of elements in the dataset
len(df_ad_budget_sales.columns)
```

[18]:
```
4
```

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_budget_sales.shape
```

[19]:
```
(200, 4)
```

[20]:
```
#View the names of each of the attributes
df_ad_budget_sales.columns
```

[20]:
```
Index(['TV Ad Budget ($)', 'Radio Ad Budget ($)', 'Newspaper Ad Budget ($)',
       'Sales ($)'],
      dtype='object')
```

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

[23]:
```
#Create a feature object from the columns
X_features = df_ad_budget_sales[['TV Ad Budget ($)', 'Radio Ad Budget ($)',␣
 ↪'Newspaper Ad Budget ($)']]
```

[24]:
```
#View the feature object
X_features
```

[24]:
```
     TV Ad Budget ($)  Radio Ad Budget ($)  Newspaper Ad Budget ($)
1               230.1                 37.8                     69.2
2                44.5                 39.3                     45.1
3                17.2                 45.9                     69.3
4               151.5                 41.3                     58.5
5               180.8                 10.8                     58.4
..                …                  …                       …
196              38.2                  3.7                     13.8
197              94.2                  4.9                      8.1
198             177.0                  9.3                      6.4
199             283.6                 42.0                     66.2
200             232.1                  8.6                      8.7

[200 rows x 3 columns]
```

[25]:
```
#Create a target object (Hint: use the sales column as it is the response of␣
 ↪the dataset)
Y_target = df_ad_budget_sales[['Sales ($)']]
```

[26]:
```
#View the target object
Y_target
```

[26]:
```
     Sales ($)
1         22.1
2         10.4
```

```
3          9.3
4         18.5
5         12.9
..         …
196        7.6
197        9.7
198       12.8
199       25.5
200       13.4

[200 rows x 1 columns]
```

[27]:

```
#Verify if all the observations have been captured in the feature object
X_features.shape
```

[27]:

```
(200, 3)
```

[28]:

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

[28]:

```
(200, 1)
```

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

[31]:

```
#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_features, Y_target,␣
 ↪random_state = 1)
```

[32]:

```
#Verify if the training and testing datasets are split correctly (Hint: use the␣
 ↪shape() method)
X_train.shape, X_test.shape, Y_train.shape, Y_test.shape
```

[32]:

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

6: Create a model to predict the sales outcome

[35]:

```
#Create a linear regression model
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(X_train, Y_train)
```

[35]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

[36]:
```
#Print the intercept and coefficients
linreg.coef_, linreg.intercept_
```

[36]:
```
(array([[0.04656457, 0.17915812, 0.00345046]]), array([2.87696662]))
```

[40]:
```
#Predict the outcome for the testing dataset
Y_pred = linreg.predict(X_test)
Y_pred
```

[40]:
```
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)

[41]:

```
#Import required libraries for calculating MSE (mean square error)

import numpy as np
from sklearn import metrics
```

[42]:

```
#Calculate the MSE
mse = np.sqrt(metrics.mean_squared_error(Y_test,Y_pred))
mse
```

[42]:

```
1.404651423032895
```