Task 3: Sales Prediction

3Q) Sales prediction means predicting how much of a product people will buy based on factors such as the amount you spend to advertise your product, the segment of people you advertise for, or the platform you are advertising on about your product. Typically, a product and service-based business always need their Data Scientist to predict their future sales with every step they take to manipulate the cost of advertising their product. So let's start the task of sales prediction with machine learning using Python.

In [1]:

```
# importing all required packages

import numpy as np
import pandas as pd
```

In [2]:

```
# reading the dataset

df=pd.read_csv('Advertising.csv')
df.head()
```

Out[2]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

In [3]:

```
df.shape
```

Out[3]:

```
(200, 5)
```

In [4]:

```
df.dtypes # identifying the datatypes of each column in the dataset
```

Out[4]:

```
Unnamed: 0      int64
TV            float64
Radio         float64
Newspaper     float64
Sales         float64
dtype: object
```

In [5]:

```
df.isnull().sum() # checking the number of null values in the data, in
```

Out[5]:

```
Unnamed: 0    0
TV            0
Radio         0
Newspaper     0
Sales         0
dtype: int64
```

In [6]:

```
df.drop('Unnamed: 0',axis=1,inplace=True) # drop id column to reduce cl
```

In [7]:

```
df.head()
```

Out[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	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

In [8]:

```
df.shape
```

Out[8]:

```
(200, 4)
```

In [9]:

```
df.dtypes
```

Out[9]:

```
TV             float64
Radio          float64
Newspaper      float64
Sales          float64
dtype: object
```

In [10]:

```
X=df.drop('Sales',axis=1) # seperating input columns from output columr
X.head() # here, input column consists of TV, Radio, Newspaper
```

Out[10]:

	TV	Radio	Newspaper
0	230.1	37.8	69.2
1	44.5	39.3	45.1
2	17.2	45.9	69.3
3	151.5	41.3	58.5
4	180.8	10.8	58.4

In [11]:

```
y=df['Sales'] # here, the output column is sales
y.head()
```

Out[11]:

```
0    22.1
1    10.4
2     9.3
3    18.5
4    12.9
Name: Sales, dtype: float64
```

In [12]:

```
# Splitting data into train and test data

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,randon
```

In [13]:

```
X_train.shape,y_train.shape
```

Out[13]:

```
((140, 3), (140,))
```

In [14]:

```
X_test.shape,y_test.shape
```

Out[14]:

```
((60, 3), (60,))
```

A) Linear Regression

In [15]:

```
# using a linear regression model

from sklearn.linear_model import LinearRegression
LR=LinearRegression()
LR.fit(X_train,y_train) # fit the train data into the model
```

Out[15]:

```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [16]:

```
y_train_pred_LR=LR.predict(X_train) # use the fit model to predict the
y_train_pred_LR
```

Out[16]:

```
array([17.39149783, 15.19196153, 11.41650701, 11.20610472, 16.39256165,
        6.90577778, 21.17740606,  6.10528574,  9.66662607, 11.67308587,
        9.0704377 ,  6.30582199, 14.7930121 , 17.42999512, 16.14599956,
       16.12273906, 14.9308629 , 19.61593142, 13.70741553, 21.09067507,
       13.09101877, 13.79551693,  8.87971636, 16.97348947,  8.22902448,
       15.34026923, 13.95862675, 23.23157581, 12.67992504, 23.10768546,
        6.80057243, 18.81832259, 23.69669553, 18.39890879, 16.97890645,
       16.44055305, 12.41657918, 11.94072527, 16.7732918 , 14.5933997 ,
       13.22910727,  7.49691601, 19.30121038,  9.33105452, 19.3603766 ,
       10.16463427,  6.94369039, 16.52918217, 13.53571009, 14.80225851,
       11.13948107, 20.85632272, 24.02985438, 18.427486  , 17.84570024,
       15.11333638, 17.21352856,  9.2345359 , 17.29735156, 19.22919752,
       16.29255016,  3.54203145,  5.24055709, 15.86268553, 15.12384811,
       16.31596188, 18.34249185, 23.4941462 , 14.34540589, 20.48784226,
       17.29893239, 21.67652708, 10.30595955, 15.25363154, 13.68052604,
        9.90613461, 18.01712723, 18.43772847, 20.3652561 , 19.37970879,
       10.18390434, 15.52162749, 20.86386386,  8.24145294, 11.54953301,
        7.25535492, 11.99074605, 14.68055076,  4.4950967 , 14.83591282,
       19.81147067, 13.60063366, 17.42635854,  3.5132451 ,  8.56886762,
       17.93522755, 14.44803773, 15.26734959, 23.38305647, 10.59905655,
       24.94180909, 12.47872281, 12.99022295, 24.44392073, 13.19055975,
       20.1322331 ,  6.91136723, 12.36791752,  7.89158996, 22.14077381,
       16.21139454, 12.18638677, 14.34672625, 12.65039181, 14.12757003,
        4.50133295,  8.02296589, 16.15950779,  9.73534671, 18.45288885,
       12.81198432, 20.82621122,  9.96179489, 10.6914528 , 10.04931249,
       14.53862883, 17.29195706, 11.86800298, 16.11235485, 17.1038322 ,
        8.20867785, 14.70885801, 18.21928267, 18.10546473, 10.61464308,
        6.20700241, 18.57459979, 19.38285038, 12.11917215, 17.21444781])
```

In [17]:

```python
# compare the y_train data and y_train_pred_LR data to check the perfor

from sklearn.metrics import r2_score,mean_absolute_error
R2=r2_score(y_train,y_train_pred_LR)
MSE=mean_absolute_error(y_train,y_train_pred_LR)
RMSE=np.sqrt(MSE)
print('R-Square: ',R2)
print('Mean Square Error: ',MSE)
print('Root Mean Square Error: ',RMSE)
```

```
R-Square:  0.9055159502227753
Mean Square Error:  1.158150294807253
Root Mean Square Error:  1.076173914758787
```

In [18]:

```python
y_test_pred_LR=LR.predict(X_test)  # use the fit model to predict the c
y_test_pred_LR
```

Out[18]:

```
array([16.5653963 , 21.18822792, 21.55107058, 10.88923816, 22.20231988,
       13.35556872, 21.19692502,  7.35028523, 13.27547079, 15.12449511,
        9.01443026,  6.52542825, 14.30205991,  8.97026042,  9.45679576,
       12.00454351,  8.91549403, 16.15619251, 10.29582883, 18.72473553,
       19.76821818, 13.77469028, 12.49638908, 21.53501762,  7.60860741,
        5.6119801 , 20.91759483, 11.80627665,  9.08076637,  8.51412012,
       12.17604891,  9.9691939 , 21.73008956, 12.77770578, 18.1011362 ,
       20.07590796, 14.26202556, 20.93826535, 10.83938827,  4.38190607,
        9.51332406, 12.40486324, 10.17045434,  8.09081363, 13.16388427,
        5.2243552 ,  9.28893833, 14.09330719,  8.69024497, 11.66119763,
       15.71848432, 11.63156862, 13.35360735, 11.1531472 ,  6.33636845,
        9.76157954,  9.4195714 , 24.25516546,  7.69519137, 12.15317572])
```

In [19]:

```python
# compare the y_test data and y_test_pred_LR data to check the performa

R2=r2_score(y_test,y_test_pred_LR)
MSE=mean_absolute_error(y_test,y_test_pred_LR)
RMSE=np.sqrt(MSE)
print('R-Square: ',R2)
print('Mean Square Error: ',MSE)
print('Root Mean Square Error: ',RMSE)
```

```
R-Square:  0.8609466508230368
Mean Square Error:  1.5116692224549084
Root Mean Square Error:  1.229499582128806
```

In [20]:

```python
LR.coef_ # this gives the coefficient of each independent variable in t
```

Out[20]:

```
array([0.04405928, 0.1992875 , 0.00688245])
```

In [21]:

```python
LR.intercept_ # this gives the intercept of the linear regression equat
```

Out[21]:

```
2.70894909251591
```

Sales = 2.70894909251591 + 0.04405928 * TV + 0.1992875 * Radio + 0.00688245 * Newspaper

B) Decision Tree

In [22]:

```
# using a linear regression model

from sklearn.tree import DecisionTreeRegressor
DT=DecisionTreeRegressor()
DT.fit(X_train,y_train)  # fit the train data into the model
```

Out[22]:

```
DecisionTreeRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [23]:

```
y_train_pred_DT=DT.predict(X_train) # use the fit model to predict the
y_train_pred_DT
```

Out[23]:

```
array([15. , 15.5, 11.9,  9.2, 12.8,  6.6, 20.7,  6.7, 11.2,  9.5, 10.7,
        8.8, 14.9, 17.1, 15.9, 15.9, 15. , 20.7, 11.7, 21.8,  9.3, 14.2,
       10.6, 17.3,  9.9, 15.2, 13.3, 25.4, 10.8, 24.2,  5.6, 19.2, 23.8,
       17.4, 17.4, 17.3, 12.9, 11.8, 15.9, 15.5, 12.9,  9.6, 20.1, 10.3,
       19.8, 11.4,  8.6, 15.7, 14.1, 13.2,  8. , 22.1, 25.5, 19. , 18.3,
       15.2, 18. ,  8.5, 18. , 18.9, 16.6,  5.3,  3.2, 15.3, 12. , 15.5,
       17.6, 25.4, 12.3, 21.5, 17.1, 23.2, 11.5, 15.6, 12.9,  9.6, 18.5,
       19.2, 21.2, 19.6, 10.5, 11.8, 22.6,  9.7, 11.8,  9.5, 13.2, 13.4,
        7.3, 13.6, 20.2, 12.2, 16.7,  4.8,  9.7, 18.4, 14.5, 12.7, 24.4,
        8.8, 27. , 10.8, 12.2, 26.2, 14. , 20.2,  8.7, 11.4,  9.7, 23.8,
       14.8, 12.9, 12.5, 13.2, 14.4,  5.9,  9.7, 14.7, 10.1, 19.6, 10.4,
       22.6, 10.1,  1.6, 11.6, 14.7, 17.2, 12.2, 16. , 17. ,  7. , 13.4,
       18. , 15.9, 12.4,  7.2, 19. , 19.4, 12.6, 14.8])
```

In [25]:

```
y_train
```

Out[25]:

```
169    15.0
97     15.5
31     11.9
12      9.2
35     12.8
       ... 
106     7.2
14     19.0
92     19.4
179    12.6
102    14.8
Name: Sales, Length: 140, dtype: float64
```

In [24]:

```
# compare the y_train data and y_train_pred_LR data to check the perfor

R2=r2_score(y_train,y_train_pred_DT)
MSE=mean_absolute_error(y_train,y_train_pred_DT)
RMSE=np.sqrt(MSE)
print('R-Square: ',R2)
print('Mean Square Error: ',MSE)
print('Root Mean Square Error: ',RMSE)
```

```
R-Square:  1.0
Mean Square Error:  0.0
Root Mean Square Error:  0.0
```

In [26]:

```
y_test_pred_DT=DT.predict(X_test) # use the fit model to predict the ou
y_test_pred_DT
```

Out[26]:

```
array([18.5, 23.8, 17.6,  5.3, 23.8, 15.3, 22.6, 10.1, 12. , 16.6,  8.8,
        8.6, 11.7,  3.2, 10.1, 12.9,  5.3, 17.3,  9.7, 20.2, 17.6, 15.3,
       10.8, 23.8,  9.9,  8.7, 22.6, 12.2, 10.6,  4.8, 11.4,  9.9, 23.8,
        9.3, 15.9, 23.8, 10.4, 17.6, 12.4,  6.7, 11.2, 12.6, 10.4,  9.9,
       11.8,  8.8,  9.7, 15. ,  9.7, 11.8, 13.6, 12.9,  5.3,  5.3,  8.8,
       11.6,  9.9, 24.4,  6.6, 11.4])
```

In [27]:

```
# compare the y_test data and y_test_pred_LR data to check the performa

R2=r2_score(y_test,y_test_pred_DT)
MSE=mean_absolute_error(y_test,y_test_pred_DT)
RMSE=np.sqrt(MSE)
print('R-Square: ',R2)
print('Mean Square Error: ',MSE)
print('Root Mean Square Error: ',RMSE)
```

```
R-Square:  0.9352062975938433
Mean Square Error:  1.0216666666666672
Root Mean Square Error:  1.010775280003754
```

C) Random Forest

In [31]:

```
# using a random forest model

from sklearn.ensemble import RandomForestRegressor
RF=RandomForestRegressor()
RF.fit(X_train,y_train)
```

Out[31]:

```
RandomForestRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [32]:

```
y_train_pred_RF=RF.predict(X_train) # use the fit model to predict the
y_train_pred_RF
```

Out[32]:

```
array([14.893, 15.507, 12.28 ,  9.131, 12.744,  6.719, 20.347,  6.714,
       11.331,  9.151, 10.694,  8.901, 14.967, 17.591, 15.983, 15.799,
       15.519, 20.476, 11.996, 22.171,  8.477, 14.445, 10.429, 17.486,
        9.97 , 14.979, 13.094, 25.347, 10.692, 24.491,  5.657, 18.993,
       23.216, 16.821, 17.272, 17.145, 13.032, 11.558, 15.742, 15.47 ,
       13.551,  9.618, 19.711, 10.457, 19.372, 11.273,  9.02 , 15.498,
       14.275, 13.192,  7.949, 22.047, 25.237, 18.941, 18.278, 15.194,
       17.46 ,  8.26 , 18.108, 19.158, 16.687,  5.442,  3.496, 14.772,
       12.162, 15.474, 17.122, 24.666, 12.264, 21.696, 17.093, 23.07 ,
       11.849, 15.62 , 12.858,  9.835, 18.307, 18.758, 20.797, 19.592,
       10.93 , 11.886, 22.289,  9.776, 11.717,  9.462, 13.054, 13.049,
        7.729, 13.384, 20.089, 12.203, 16.552,  4.515,  9.819, 18.322,
       14.712, 12.363, 24.126,  9.012, 26.413, 11.174, 12.187, 25.901,
       13.295, 20.22 ,  9.007, 11.74 ,  9.62 , 23.223, 15.028, 12.351,
       12.387, 13.389, 14.312,  6.218,  9.848, 14.427, 10.376, 19.34 ,
       10.667, 22.156, 10.062,  3.983, 11.723, 14.824, 17.465, 12.501,
       15.896, 17.114,  7.232, 13.28 , 18.413, 15.555, 12.233,  7.524,
       18.83 , 19.381, 12.931, 14.437])
```

In [33]:

```
# compare the y_train data and y_train_pred_LR data to check the perfor

R2=r2_score(y_train,y_train_pred_RF)
MSE=mean_absolute_error(y_train,y_train_pred_RF)
RMSE=np.sqrt(MSE)
print('R-Square: ',R2)
print('Mean Square Error: ',MSE)
print('Root Mean Square Error: ',RMSE)
```

```
R-Square:  0.9954046126694128
Mean Square Error:  0.23939285714285685
Root Mean Square Error:  0.48927789357670437
```

In [34]:

```
y_test_pred_RF=RF.predict(X_test) # use the fit model to predict the ou
y_test_pred_RF
```

Out[34]:

```
array([17.455, 22.02 , 20.401,  6.315, 23.268, 13.091, 22.686,  9.715,
       12.361, 16.106,  8.421,  9.056, 12.191,  4.562, 10.532, 12.465,
        5.816, 16.739, 11.095, 19.596, 19.944, 12.988, 10.614, 22.108,
        9.97 ,  8.994, 22.32 , 12.536, 10.18 ,  5.126, 11.558, 10.838,
       22.384,  8.477, 15.199, 20.349, 11.954, 20.515, 12.416,  7.585,
       11.574, 13.004, 10.357,  9.698, 11.996,  9.013, 10.726, 15.508,
       10.796, 11.842, 13.651, 12.604,  6.404,  5.936,  9.002, 11.409,
       10.763, 25.265,  6.827, 12.227])
```

In [35]:

```python
# compare the y_train data and y_train_pred_LR data to check the perfor

R2=r2_score(y_test,y_test_pred_RF)
MSE=mean_absolute_error(y_test,y_test_pred_RF)
RMSE=np.sqrt(MSE)
print('R-Square: ',R2)
print('Mean Square Error: ',MSE)
print('Root Mean Square Error: ',RMSE)
```

```
R-Square:  0.9830724032237538
Mean Square Error:  0.5809333333333325
Root Mean Square Error:  0.7621898276238883
```

In [ ]:

```

```