In [1]:

```
import pandas as pd
df = pd.read_csv("C:\\Users\\Sushmitha T\\Downloads\\Advertising Dataset.csv")
```

In [8]:

```
feature_cols=['TV','Radio','Newspaper']
```

In [10]:

```
x=df[feature_cols]
x=df[['TV','Radio','Newspaper']]
x.head()
```

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]:

```
print(type(x))
print(x.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
(200, 3)
```

In [13]:

```
y=df['Sales']
y=df.Sales
y.head()
```

Out[13]:

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

In [14]:

```
print(type(y))
print(y.shape)
```

```
<class 'pandas.core.series.Series'>
(200,)
```

Splitting x and y training and testing sets

In [22]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=1)
```

In [23]:

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

Linear Regression in scikit-learn

In [24]:

```
from sklearn.linear_model import LinearRegression
```

In [26]:

```
linreg=LinearRegression()
linreg.fit(x_train,y_train)
```

Out[26]:

```
▾ LinearRegression ⓘ ?

LinearRegression()
```

Interpreting model coefficients

In [27]:

```
print(linreg.intercept_)
print(linreg.coef_)
```

```
2.8769666223179335
[0.04656457 0.17915812 0.00345046]
```

In [28]:

```
list(zip(feature_cols,linreg.coef_))
```

```
Out[28]: [('TV', np.float64(0.046564567874150295)),
          ('Radio', np.float64(0.1791581224508883)),
          ('Newspaper', np.float64(0.003450464711180402))]
```

In [29]:

```
y_pred=linreg.predict(x_test)
```

In [30]:

```
true=[100,50,30,20]
pred=[90,50,50,30]
```

In [17]:

```
from sklearn import metrics
true = [100,50,30,20]
pred = [90,50,50,30]
print((10+0+20+10)/4.)
print(metrics.mean_absolute_error(true, pred))
```

```
10.0
10.0
```

In [18]:

```
print((10**2 + 0**2 + 20**2 + 10**2)/4.)
from sklearn import metrics
print(metrics.mean_squared_error(true,pred))
```

```
150.0
150.0
```

central tendancy

In [19]:

```
mean_values = df.mean()
print("Mean:\n", mean_values)
```

```
Mean:
 Unnamed: 0    100.5000
TV            147.0425
Radio          23.2640
Newspaper      30.5540
Sales          14.0225
dtype: float64
```

In [20]:

```
median_values = df.median()
print("Median:\n", median_values)
```

```
Median:
 Unnamed: 0    100.50
TV            149.75
Radio          22.90
Newspaper      25.75
Sales          12.90
dtype: float64
```

In [21]:

```
mode_values = df.mode()
print("Mode:\n", mode_values)
```

```
Mode:
      Unnamed: 0     TV  Radio  Newspaper  Sales
0             1   17.2    4.1        8.7    9.7
1             2   76.4    5.7        9.3    NaN
2             3  109.8    NaN       25.6    NaN
3             4  177.0    NaN        NaN    NaN
4             5  184.9    NaN        NaN    NaN
..          ...    ...    ...        ...    ...
195         196    NaN    NaN        NaN    NaN
196         197    NaN    NaN        NaN    NaN
197         198    NaN    NaN        NaN    NaN
198         199    NaN    NaN        NaN    NaN
199         200    NaN    NaN        NaN    NaN

[200 rows x 5 columns]
```

Measures of Dispersion

In [22]:

```
range_values = df.max() - df.min()
print("Range:\n", range_values)
```

```
Range:
 Unnamed: 0    199.0
TV            295.7
Radio          49.6
Newspaper     113.7
Sales          25.4
dtype: float64
```

In [23]:

```
variance_values = df.var()
print("Variance:\n", variance_values)
```

```
Variance:
 Unnamed: 0    3350.000000
TV            7370.949893
Radio          220.427743
Newspaper      474.308326
Sales           27.221853
dtype: float64
```

In [24]:

```
std_dev_values = df.std()
print("Standard Deviation:\n", std_dev_values)
```

```
Standard Deviation:
 Unnamed: 0    57.879185
TV            85.854236
Radio         14.846809
Newspaper     21.778621
Sales          5.217457
dtype: float64
```

In [ ]:

Identifying the intercept (c) and coefficients ($m1$,$m2$,$m3$,…):

In [23]:

```
X=df.drop('Sales',axis=1)
y=df['Sales']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
intercept = model.intercept_
coefficients = model.coef_

# Print the intercept and coefficients
print(f"Intercept (c): {intercept}")
print(f"Coefficients (m1, m2, m3, ...): {coefficients}")
```

```
Intercept (c): 2.906527086361816
Coefficients (m1, m2, m3, ...): [0.00064359 0.04471835 0.18925118 0.00304577]
```

In [ ]:

In [25]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
data = pd.read_csv('C:\\Users\\Sushmitha T\\Downloads\\Advertising Dataset.csv')
X = data.drop('Sales', axis=1)
Y = data['Sales']
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,random_state=42)
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(X_train, Y_train)

Y_pred = linreg.predict(X_test)

mse = mean_squared_error(Y_test, Y_pred)
r2 = r2_score(Y_test, Y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared (R²) score: {r2}")

comparison = pd.DataFrame({'Actual': Y_test, 'Predicted': Y_pred})
print(comparison.head())
```

```
Mean Squared Error: 3.199004468588908
R-squared (R²) score: 0.8986489151417079
     Actual  Predicted
95     16.9  16.412277
15     22.4  20.843193
30     21.4  21.511869
158     7.3  10.653100
128    24.7  22.124058
```

In [27]:

```
import numpy as np
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(Y_test, Y_pred))
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R²) score: {r2}")
print("\n")
```

```
Root Mean Squared Error (RMSE): 1.7885761008659677
R-squared (R²) score: 0.8986489151417079
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js