Predicting Revenue of an Ice Cream Shop depending upon the Temperature.

So we have a dataset of a Ice Cream Shop wherein

- "Temperature" is independent variable
- "Revenue" is dependent variable

So we're going to build a **Decision Tree Regressor** to find the relation between these two variables.

Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeRegressor
```

Import the Dataset

```
df = pd.read_csv('https://raw.githubusercontent.com/mk-gurucharan/Regression/master/IceCreamD
```

```
df.head()
```

	Temperature	Revenue
0	24.566884	534.799028
1	26.005191	625.190122
2	27.790554	660.632289
3	20.595335	487.706960
4	11.503498	316.240194

```
df.describe()
```

	Temperature	Revenue
count	500.000000	500.000000
mean	22.232225	521.570777

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Temperature  500 non-null    float64
 1   Revenue      500 non-null    float64
dtypes: float64(2)
memory usage: 7.9 KB
```

To check whether we have Missing Value

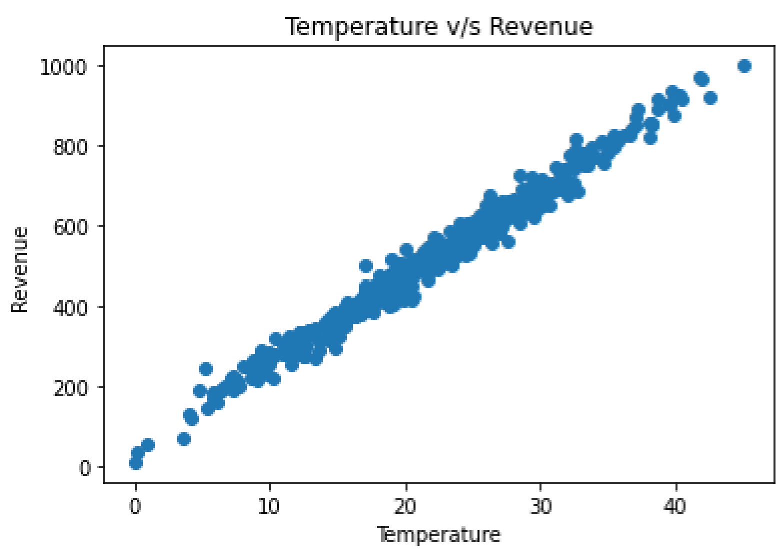
```
df.isnull().sum().sum()
```

```
0
```

Data Visualization

```
plt.scatter(df.Temperature,df.Revenue)
plt.xlabel('Temperature')
plt.ylabel('Revenue')
plt.title('Temperature v/s Revenue')
```

```
Text(0.5, 1.0, 'Temperature v/s Revenue')
```

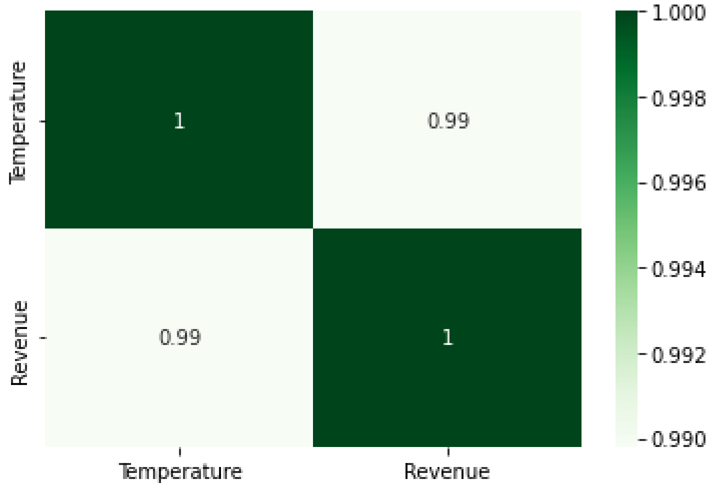


this clearly shows that there is a linear relationship between the two; hence we'll make a simple Linear Regression model

Validating the correlation matrix using Heatmap

```
sns.heatmap(df.corr(), annot=True, cmap='Greens')
```

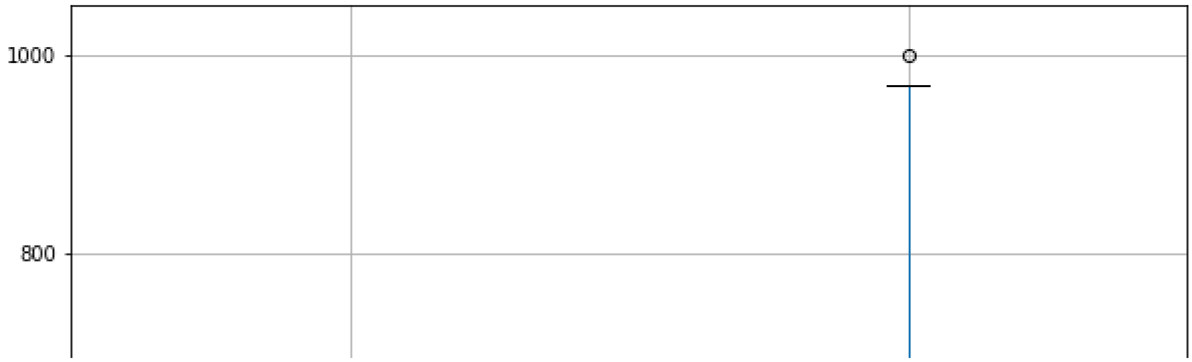
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6fefc0950>
```



Check the outliers

```
plt.figure(figsize=(10,10))
df.boxplot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6e6d40e50>
```



Since there are 3 to 4 outliers we can move ahead with it.

Feature Scaling

Splitting the Data for Training and Testing

```
x=np.array(df.Temperature.values)

y= np.array(df.Revenue.values)
```

```
from sklearn.preprocessing import StandardScaler
stanscale = StandardScaler()
x=stanscale.fit_transform(x.reshape(-1, 1))
y=stanscale.fit_transform(y.reshape(-1, 1))
```

```
x
```

```
       [-6.30470911e-01],
       [-1.66140519e+00],
       [ 1.08775907e+00],
       [-3.74523684e-01],
       [ 6.05466483e-01],
       [ 9.70398816e-01],
       [ 2.81490687e+00],
       [-4.10932058e-01],
       [-8.81435478e-01],
       [-4.13922552e-02],
       [ 8.99568644e-01],
       [-3.65164265e-01],
       [ 9.45533668e-01],
       [-7.68495660e-02],
       [ 3.40775969e-01],
       [ 1.10379429e+00],
       [ 1.53874624e+00],
       [-1.64639350e-01],
       [ 1.84153271e+00],
       [ 5.10404179e-01],
       [ 1.61950026e+00],
```

```
       [ 2.78511813e-01],
       [ 2.03207146e+00],
       [ 8.24397864e-01],
       [-1.24985809e+00],
       [-1.31512671e+00],
       [-5.90614976e-01],
       [-1.65128400e-01],
       [ 2.23428551e+00],
       [ 5.31384903e-01],
       [ 2.13653762e+00],
       [ 2.04928639e-02],
       [-1.59772760e+00],
       [-3.38450145e-01],
       [-8.78162869e-04],
       [-4.13568834e-01],
       [-5.93615097e-02],
       [-5.03568715e-01],
       [-9.10714661e-01],
       [ 8.15689884e-01],
       [ 3.83418021e-01],
       [-4.60402144e-01],
       [ 3.09797555e-02],
       [ 9.70905758e-01],
       [-6.47150147e-01],
       [ 6.24216061e-01],
       [-2.14709880e+00],
       [ 1.45275735e-01],
       [-1.22777667e+00],
       [ 1.28588821e+00],
       [-6.83503593e-01],
       [ 5.85042583e-01],
       [ 1.96914036e-01],
       [ 1.51329248e+00],
       [ 1.01874284e-01],
       [-9.02628610e-01],
       [ 3.56050740e-01],
       [ 5.27604251e-03],
```

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
```

Using Decision Tree Regressor Model

```
regressor = DecisionTreeRegressor()
```

Train the model

```
regressor.fit(x_train,y_train)
```

```
DecisionTreeRegressor()
```

Making Predictions and Checking Accuracy

```
ypred = regressor.predict(x_test)
```

```
ypred
```

```
array([ 7.56874910e-01, -2.86605995e-02,  2.91625218e-01,  1.78329663e-01,
       -4.13503479e-01, -1.76390188e+00,  8.80321817e-01,  1.75042286e+00,
        4.73984178e-01, -6.85603304e-01, -1.82627531e-01, -2.79755986e-01,
       -6.63818309e-01, -2.45234549e-01, -1.72497896e+00, -1.13329704e+00,
       -7.85339951e-01,  5.95081241e-01, -1.70461521e+00, -4.00161440e-01,
        7.11617285e-01, -1.13977828e+00, -1.02557886e+00, -2.73735347e-01,
       -1.13329704e+00, -1.91444712e-02,  5.23739500e-02,  1.64758416e-01,
        9.60347123e-01, -1.00943195e+00, -3.33877818e-01,  9.04280172e-01,
        1.16800419e-03, -1.72497896e+00,  7.98364060e-01, -5.49595324e-01,
        1.56906963e+00,  2.79674177e-01, -1.91231928e+00,  5.95081241e-01,
       -6.40121269e-01,  1.01239389e+00,  3.58554096e-01,  7.55672655e-01,
        1.03534784e+00,  6.98362540e-01, -3.93452899e-01, -1.54905095e+00,
       -6.24928606e-01,  1.80043419e-01,  9.65697857e-01,  6.81704720e-01,
       -1.13977828e+00,  1.99279015e+00, -7.80149093e-02,  6.09957692e-01,
       -1.63332389e+00, -1.37563961e+00,  8.96999807e-01, -1.43294644e-01,
       -4.56898833e-01,  4.92317424e-01,  3.73537129e-02,  1.75042286e+00,
        4.43956321e-01,  1.20781799e+00,  1.38864666e-01, -1.17821147e-01,
        7.44365597e-01, -1.29105767e+00,  1.03534784e+00, -8.10404099e-01,
        1.03323405e+00,  5.72954729e-01,  9.65697857e-01,  1.44118230e+00,
       -4.13503479e-01, -6.24928606e-01, -6.03512811e-01,  5.95081241e-01,
        2.74200667e-01,  4.96830360e-01, -4.37963247e-01, -7.27059843e-01,
        4.26104146e-01, -1.13552975e+00,  7.82002281e-01,  2.19088314e+00,
        1.64413911e+00,  4.43956321e-01, -1.33074983e-01, -4.37963247e-01,
        1.80043419e-01, -3.84533440e-01, -1.29912314e+00, -8.47918736e-01,
        1.64231559e+00, -1.37936228e+00,  7.82002281e-01, -2.79076772e+00])
```

```
plt.scatter(x_test,y_test, color='red')
plt.scatter(x_test,ypred, color='green')
plt.xlabel('xtest')
plt.ylabel('actual_red/pred/green')
```

```
Text(0, 0.5, 'actual_red/pred/green')
```

```
from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error
```

```
r2_score(y_test,ypred)
```

```
0.9621179075515762
```

Predictions are 96.35% accurate.

xtest

For Better Accuracy let's try Linear Regression

```
from sklearn.linear_model import LinearRegression
model1 = LinearRegression()
```

```
model1.fit(x_train, y_train)
```

```
LinearRegression()
```

```
y_pred = model1.predict(x_test)
```

```
r2_score(y_test, y_pred)
```

```
0.981279705545525
```

Predictions are 98.37% accurate.

0s completed at 5:09 PM