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/IceCreamData.csv')
```

```
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

```
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
```

75% 27.740674 642.257922

To check whether we have Missing Value

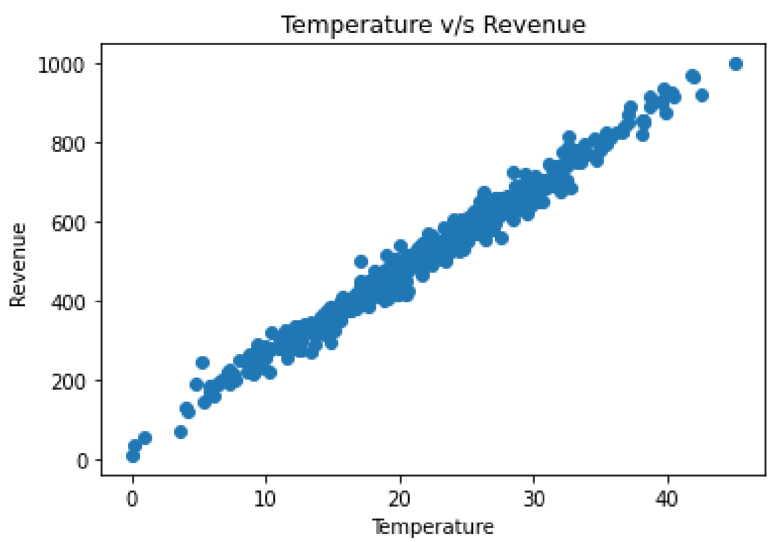
```
df.isnull().sum().sum()
```

```
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')
```

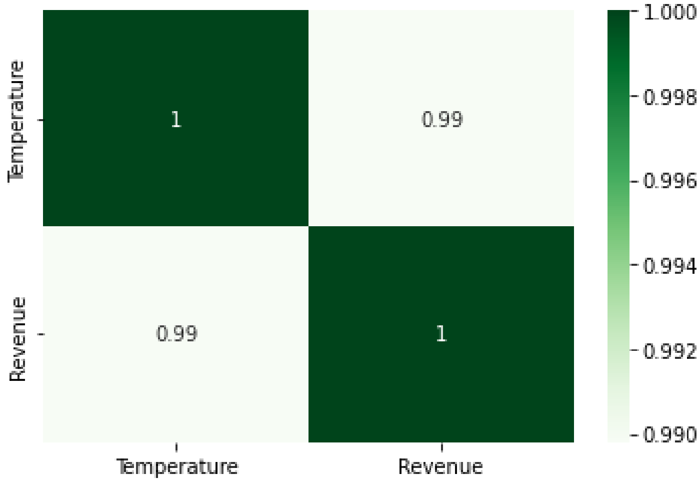


this clearly shows that there is a linear relationship between the two; hence we'll make a simple Linear Regression model

Validating the correlation matrix using Heatmap

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

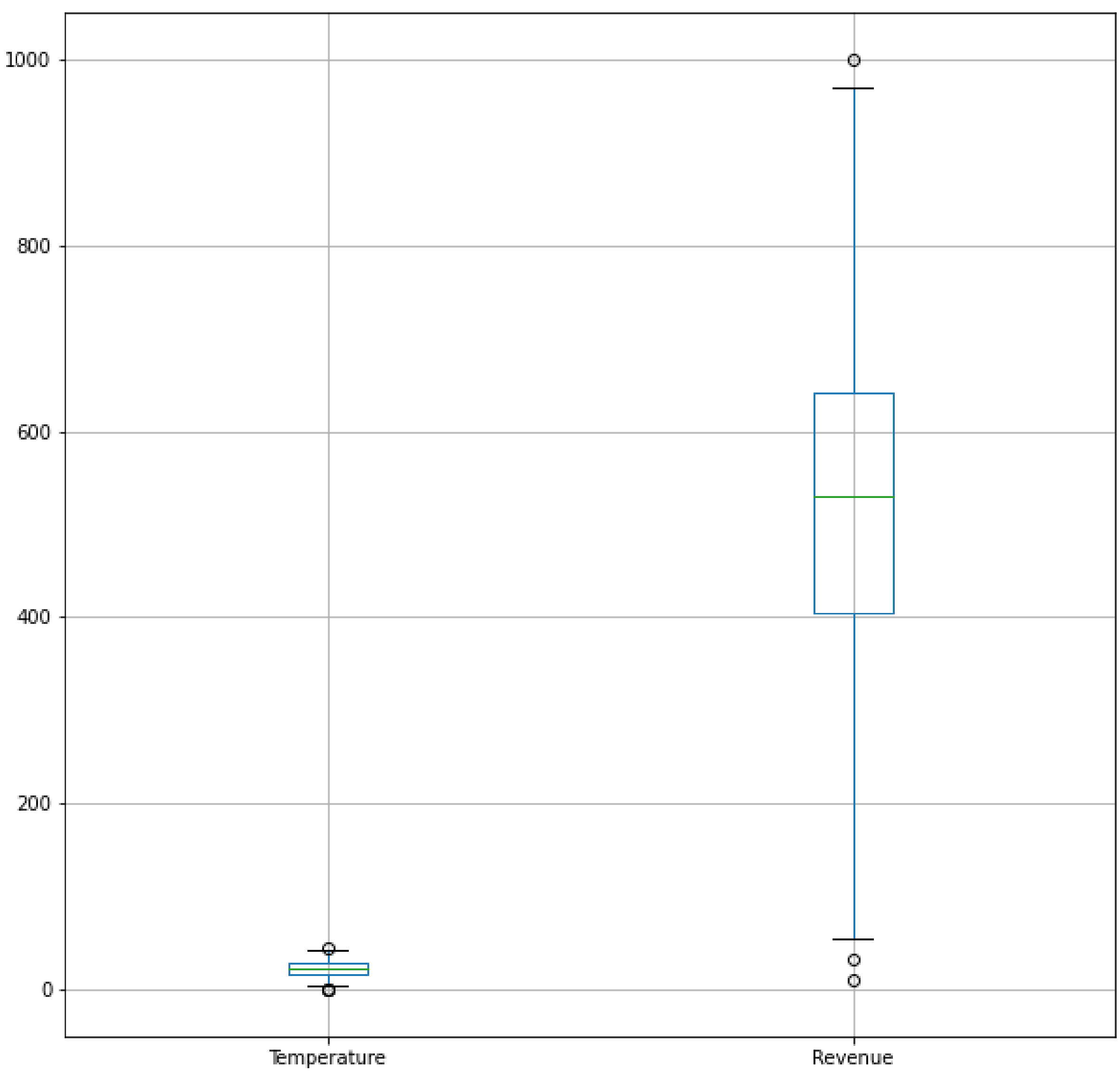
```
<matplotlib.axes._subplots.AxesSubplot at 0x7feab3a8fc90>
```



Check the outliers

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feaab4d8190>
```



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
```

```
            [-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],
```

```
       [-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],
       [ 1.31806239e+00],
```

```
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)
```

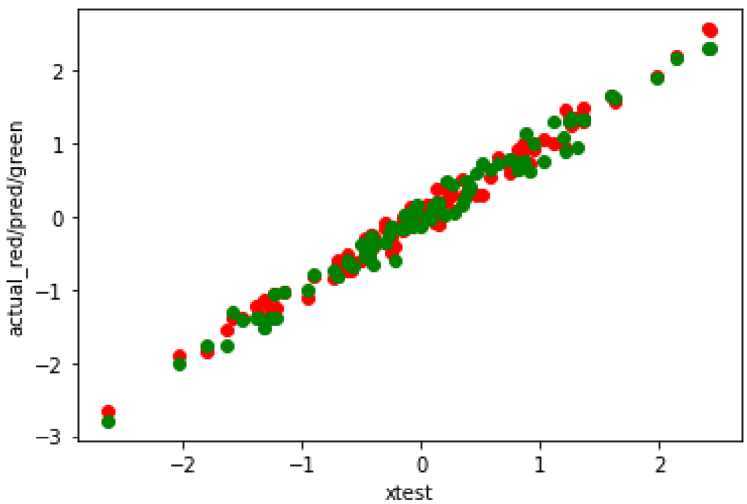
```
ypred
```

```
array([ 0.76001397,  0.93639544, -0.09032116,  0.99908772, -0.35787329,
       -0.60351281,  0.63878163,  0.00886707,  0.03735371,  0.39720886,
       -0.78109879,  0.7639318 , -0.62482956,  0.76001397, -0.25255822,
       -2.00499063,  0.16624294,  0.23786102,  0.73137498, -0.14329464,
        0.61817881, -0.35787329,  0.18004342, -0.59474051, -1.04127128,
        1.30614007,  0.12571426,  0.76001397,  1.29197884,  0.02927641,
       -1.76328451, -0.01914447, -0.17314559,  1.6132537 ,  1.12938288,
       -0.38683705,  0.71161729, -0.17314559, -0.81180821,  2.28236712,
       -1.38583486, -0.57039146, -1.50940546, -0.53849122,  0.74354455,
        0.47398418, -0.17314559,  2.15280627, -2.79076772,  0.76001397,
        0.05013967,  0.68170472,  1.8756925 , -1.37936228, -1.31169725,
       -0.26455298, -0.38453344, -0.72705984,  0.02927641,  1.30614007,
```

```
        0.12571426,  0.7639318 , -1.38762507,  0.058048  ,  0.59133586,
       -1.38762507,  0.16475842, -0.65659518, -1.01699838,  1.30614007,
        0.64494922, -0.17314559,  0.18004342,  1.08202985,  0.41511845,
       -0.01914447,  0.63878163, -0.81180821,  0.46603042,  1.64413911,
       -1.37563961,  0.18004342, -0.67784116, -1.41936203, -0.13307498,
        0.88019761, -0.14585954,  1.28617569, -1.06382874,  0.18004342,
       -0.0286606 , -0.53849122, -0.66147247, -1.76390188, -0.17314559,
       -0.43796325,  1.35711728,  2.28236712, -0.13307498,  0.35104681])
```

```
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)
```

```
0.9635471732142511
```

Predictions are 96.35% accurate.

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)
```

```
0.9845290468141842
```

Predictions are 98.37% accurate.

0s completed at 16:59