Ice-cream Sales Revenue Prediction

In [1]:
```
# import library
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# import dataset
icecream = pd.read_csv('https://github.com/YBIFoundation/Dataset/raw/ma
```

In [3]:
```
icecream.head()
```

Out[3]:

	Temperature	Revenue
0	24.6	535
1	26.1	626
2	27.8	661
3	20.6	488
4	11.6	317

In [4]:
```
icecream.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    int64  
dtypes: float64(1), int64(1)
memory usage: 7.9 KB
```

In [5]:
```
icecream.describe()
```

Out[5]:

	Temperature	Revenue
count	500.000000	500.000000
mean	22.281600	522.058000
std	8.097597	175.410399
min	0.000000	10.000000
25%	17.175000	406.000000
50%	22.400000	530.000000
75%	27.800000	643.000000
max	45.000000	1000.000000

In [6]:
```
# define y and X
icecream.columns
```

Out[6]:
```
Index(['Temperature', 'Revenue'], dtype='object')
```

In [7]:
```
y = icecream['Revenue']
X = icecream[['Temperature']]
```

In [8]:
```
# split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=2529)
```

In [9]:
```
# select model
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
```

In [10]:
```
# train model
reg.fit(X_train,y_train)
```

Out[10]:
```
LinearRegression()
```

In [11]:
```
# intercept and slope
reg.intercept_, reg.coef_
```

Out[11]:
```
(44.0180975902457, array([21.4695184]))
```

In [12]:
```
# prediction
y_pred = reg.predict(X_test)
```

In [13]:
```
# model evaluation
from sklearn.metrics import mean_absolute_error, mean_squared_error, me
```

In [14]:
```
mean_absolute_error(y_test,y_pred)
```

Out[14]:
```
18.369558283712276
```

In [15]:
```
mean_squared_error(y_test,y_pred)
```

Out[15]:
```
517.497849367057
```

In [16]:
```
mean_absolute_percentage_error(y_test,y_pred)
```

Out[16]:
```
0.041552092681591525
```

In [17]:
```
r2_score(y_test,y_pred)
```

Out[17]:
```
0.9841801999698918
```

Use Statsmodels Library

In [18]:
```
import statsmodels.api as sm
X1 = sm.add_constant(X)
print(sm.OLS(y,X).fit().summary())
```

```
                                 OLS Regression Results
=======================================================================
=================
Dep. Variable:                Revenue   R-squared (uncentered):
0.997
Model:                            OLS   Adj. R-squared (uncentered):
0.997
Method:                 Least Squares   F-statistic:
1.777e+05
Date:                Fri, 03 May 2024   Prob (F-statistic):
0.00
Time:                        14:37:49   Log-Likelihood:
-2395.6
No. Observations:                 500   AIC:
4793.
Df Residuals:                     499   BIC:
4797.
Df Model:                           1
Covariance Type:            nonrobust
=======================================================================
=========
                  coef    std err          t      P>|t|      [0.025
0.975]
-----------------------------------------------------------------------
---------
Temperature    23.1985      0.055    421.521      0.000      23.090
23.307
=======================================================================
========
Omnibus:                        3.200   Durbin-Watson:
2.020
Prob(Omnibus):                  0.202   Jarque-Bera (JB):
3.420
Skew:                           0.075   Prob(JB):
0.181
Kurtosis:                       3.377   Cond. No.
1.00
=======================================================================
========

Notes:
[1] R² is computed without centering (uncentered) since the model does
not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

In [ ]:
```
```