**WORKSHEET 2**

**MACHINE LEARNING**

**22PW28**

**PRATHOSHINI DEVI M B**

**PPT PROBLEM SOLUTION**

**import numpy as np**

**x=np.array([[1,4.0],**

**[1,4.5],**

**[1,5.0],**

**[1,5.5],**

**[1,6.0],**

**[1,6.5],**

**[1,7.0]])**

**y=np.array([[33],[42],[45],[51],[53],[61],[62]])**

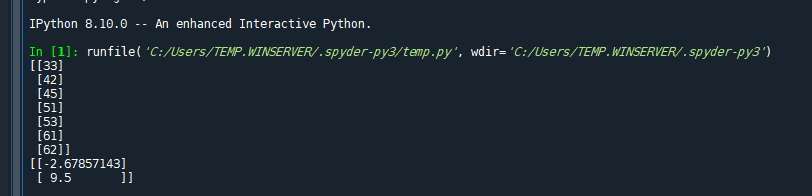
**print(y)**

**XtransposeX=np.dot(np.transpose(x),x)**

**XtransposeY=np.dot(np.transpose(x),y)**

**b=np.dot(np.linalg.inv(XtransposeX),XtransposeY)**

**print(b)**



CHANGING X

**import numpy as np**

**x=np.array([[1,45,48],**

**[1,46,49],**

**[1,27,30],**

**[1,58,41],**

**[1,52,2],**

**[1,42,18],**

**[1,25,26]])**

**y=np.array([[33],[42],[45],[51],[53],[61],[62]])**

**print(y)**

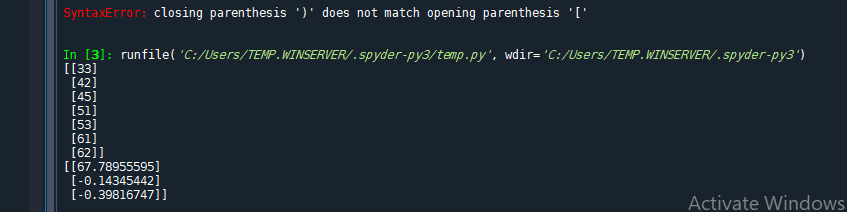
**XtransposeX=np.dot(np.transpose(x),x)**

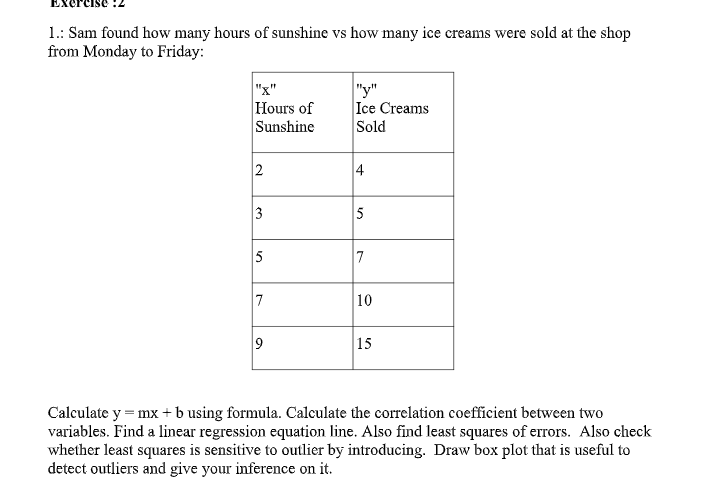
**XtransposeY=np.dot(np.transpose(x),y)**

**b=np.dot(np.linalg.inv(XtransposeX),XtransposeY)**

**print(b)**

OUTPUT:





import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Original data

x = np.array([[1, 2],

[1, 3],

[1, 5],

[1, 7],

[1, 9]])

y = np.array([[4], [5], [7], [10], [15]])

Y = np.array([4, 5, 7, 10, 15])

X = np.array([2, 3, 5, 7, 9])

# Linear regression using matrix operations

XtransposeX = np.dot(np.transpose(x), x)

Xtransposey = np.dot(np.transpose(x), y)

b = np.dot(np.linalg.inv(XtransposeX), Xtransposey)

print(b)

print('THE LINEAR REGRESSION EQUATION IS y=', b[0], '+', b[1], 'x')

# Correlation coefficient

r = np.corrcoef(X, Y)

print(r[0, 1], ' is the correlation coefficient')

# Linear regression using formula

sum\_x = sum(X)

sum\_y = sum(Y)

sum\_xy = sum(X \* Y)

sum\_xsq = sum(X \*\* 2)

b\_formula = ((5 \* sum\_xy) - (sum\_x \* sum\_y)) / ((5 \* sum\_xsq) - (sum\_x \*\* 2))

a\_formula = (sum\_y - (b\_formula \* sum\_x)) / 5

print(b\_formula)

print(a\_formula)

print('THE LINEAR REGRESSION EQUATION IS y=', a\_formula, '+', b\_formula, 'x')

# Mean Squared Error (MSE)

error = 0

for i in range(5):

error += (Y[i] - (a\_formula + b\_formula \* X[i])) \*\* 2

print('MSE', error)

# Introduce an outlier

Y\_outlier = np.array([4, 5, 7, 10, 50]) # Changed the last value to 50 to introduce an outlier

sum\_y\_outlier = sum(Y\_outlier)

sum\_xy\_outlier = sum(X \* Y\_outlier)

b\_formula\_outlier = ((5 \* sum\_xy\_outlier) - (sum\_x \* sum\_y\_outlier)) / ((5 \* sum\_xsq) - (sum\_x \*\* 2))

a\_formula\_outlier = (sum\_y\_outlier - (b\_formula\_outlier \* sum\_x)) / 5

print('With outlier:')

print(b\_formula\_outlier)

print(a\_formula\_outlier)

print('THE LINEAR REGRESSION EQUATION IS y=', a\_formula\_outlier, '+', b\_formula\_outlier, 'x')

# Mean Squared Error (MSE) with outlier

error\_outlier = 0

for i in range(5):

error\_outlier += (Y\_outlier[i] - (a\_formula\_outlier + b\_formula\_outlier \* X[i])) \*\* 2

print('MSE with outlier', error\_outlier)

# Box plot to detect outliers

plt.figure(figsize=(10, 6))

sns.boxplot(data=[Y, Y\_outlier], palette="Set3")

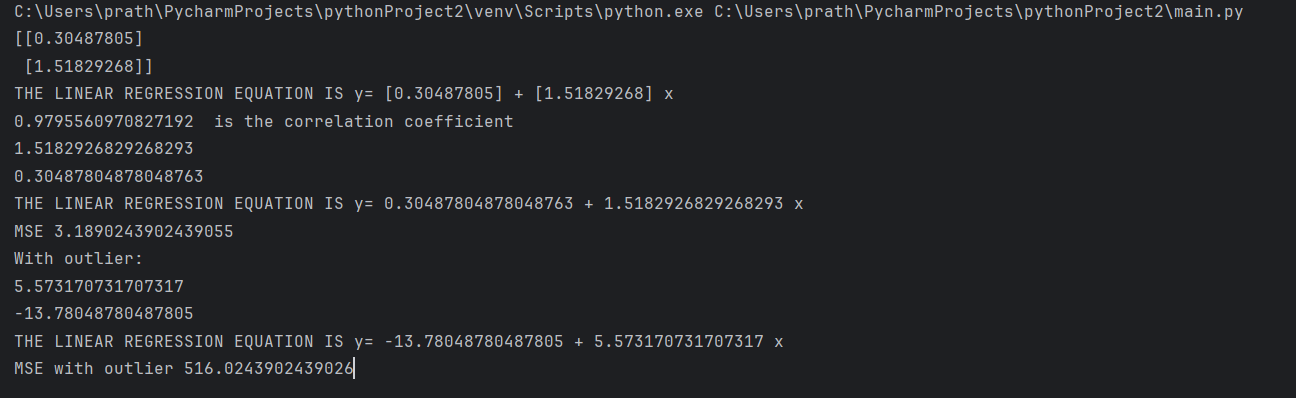
plt.xticks([0, 1], ['Original Data', 'Data with Outlier'])

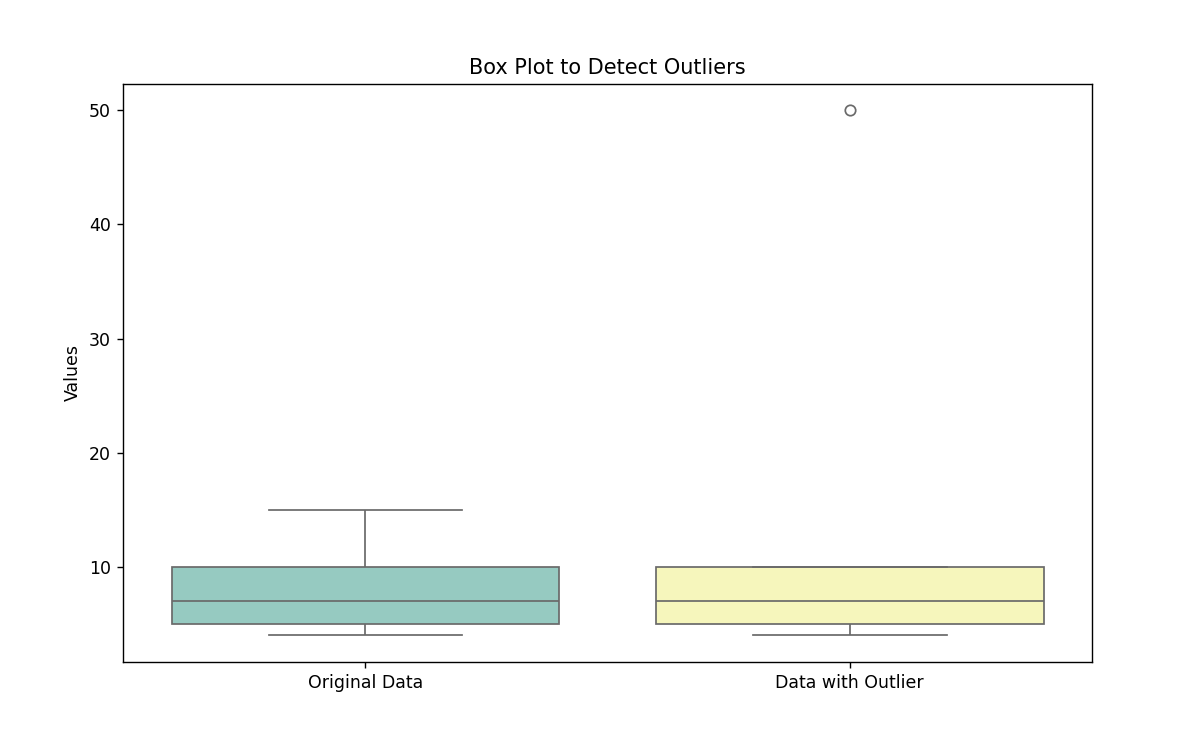
plt.title('Box Plot to Detect Outliers')

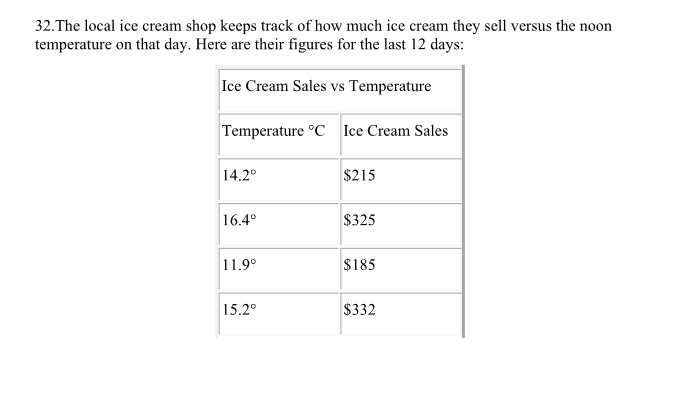
plt.ylabel('Values')

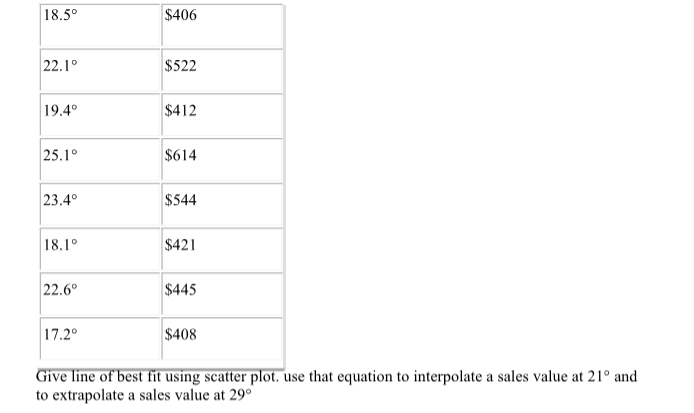
plt.show()

**OUTPUT:**

****

****





**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**from sklearn.linear\_model import LinearRegression**

**data={**

**'Temperature(C)':[14.2,16.4,11.9,15.2,18.5,22.1,19.4,25.1,23.4,18.1,22.6,17.2],**

**'Ice cream sales($)':[215,325,185,332,406,522,412,614,544,421,445,408]}**

**df=pd.DataFrame(data)**

**x=df[['Temperature(C)']]**

**y=df['Ice cream sales($)']**

**print(x)**

**print(y)**

**best\_fit\_line=LinearRegression().fit(x,y)**

**plt.scatter(x,y)**

**plt.plot(x,best\_fit\_line.predict(x))**

**plt.legend()**

**plt.show()**

**interpolate=21**

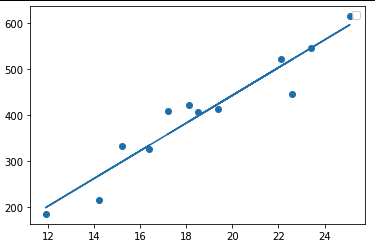
**interpolate\_result=best\_fit\_line.predict([[interpolate]])**

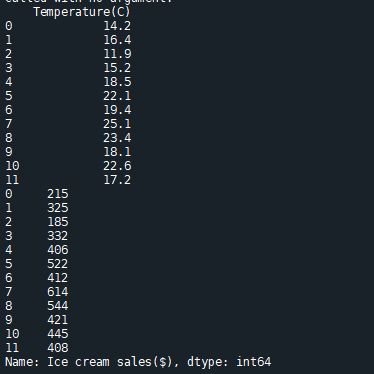
**print(f"The value of interpolation of 21 : {interpolate\_result[0]:.2f}")**

**extrapolate=29**

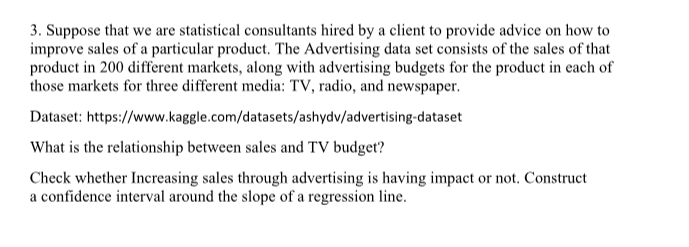
**extrapolate\_result=best\_fit\_line.predict([[extrapolate]])**

**print(f"The value of extrapolation of 29 : {extrapolate\_result[0]:.2f}")**

****

****

****



import pandas as pd

import numpy as np

from scipy import stats

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

df = pd.read\_csv(r"C:\Users\prath\Downloads\archive\advertising.csv")

print(df.columns)

tv\_col = 'TV'

sales\_col = 'Sales'

correlation = df[tv\_col].corr(df[sales\_col])

print(f"Correlation coefficient between sales and TV budget: {correlation:.2f}")

X = df[[tv\_col]]

y = df[sales\_col]

reg = LinearRegression().fit(X, y)

plt.figure(figsize=(10, 6))

plt.scatter(X, y, color='blue', label='Data points')

plt.plot(X, reg.predict(X), color='red', label='Line of best fit')

plt.xlabel('TV Budget ($)')

plt.ylabel('Sales ($)')

plt.title('Sales vs. TV Budget')

plt.legend()

plt.show()

print(f"For every $1 increase in TV budget, sales are expected to increase by ${m:.2f}")

confidence\_level = 0.95

t\_value = stats.t.ppf((1 + confidence\_level) / 2, len(X) - 2)

margin\_error = t\_value \* np.std(y) / np.sqrt(len(X))

print(f"Confidence interval for the slope: ({m - margin\_error:.2f}, {m + margin\_error:.2f})")

if m - margin\_error > 0 and m + margin\_error > 0:

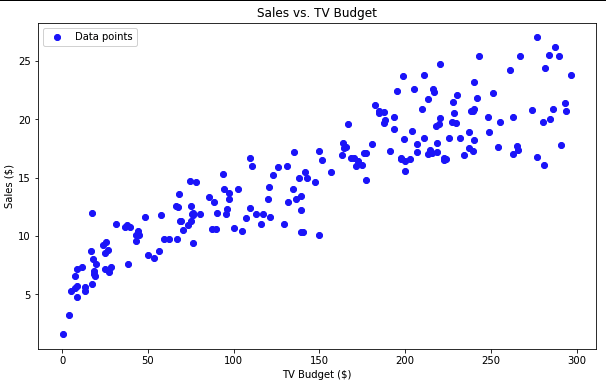
print("Increasing sales through advertising has a positive impact on sales.")

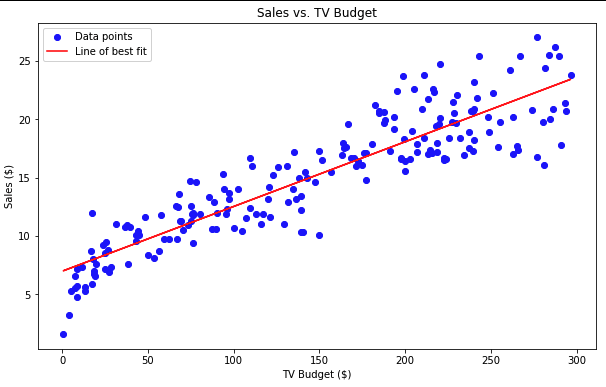
elif m - margin\_error < 0 and m + margin\_error < 0:

print("Increasing sales through advertising has a negative impact on sales.")

else:

print("We are not confident that increasing sales through advertising has a significant impact on sales.")

****

****

**Correlation coefficient between sales and TV budget: 0.90**

**Line of best fit equation: y = 0.06x + 6.97**

**For every $1 increase in TV budget, sales are expected to increase by $0.06**

**Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')**

**Correlation coefficient between sales and TV budget: 0.90**