import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.linear\_model import LinearRegression

import statsmodels.api as sm # For p-values and detailed regression stats

# Data from your Excel sheet (adjusted consumption and disposable income)

data = {

'Year': [2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023],

'Consumption': [233490, 229393, 272634, 272984, 293991, 305353, 298826, 279308, 342867, 297599],

'Disposable\_Income': [8574640, 9368020, 10284700, 11426100, 12668100, 14281400, 15364300, 15700300, 17835700, 20137900]

}

# Create DataFrame

df = pd.DataFrame(data)

# 1. Calculate Key Metrics

df['APC'] = df['Consumption'] / df['Disposable\_Income'] # Average Propensity to Consume

df['Consumption\_Income\_Ratio'] = df['Consumption'] / df['Disposable\_Income'] \* 100

# 2. Linear Regression with p-values (using statsmodels)

X = sm.add\_constant(df['Disposable\_Income']) # Adds intercept term

y = df['Consumption']

model\_sm = sm.OLS(y, X).fit() # Ordinary Least Squares regression

# Extract regression coefficients and p-values

slope = model\_sm.params['Disposable\_Income']

intercept = model\_sm.params['const']

p\_value = model\_sm.pvalues['Disposable\_Income']

r\_squared = model\_sm.rsquared

# Predictions for regression line

df['Predicted\_Consumption'] = model\_sm.predict(X)

# 3. Plotting

plt.figure(figsize=(12, 6))

sns.scatterplot(x='Disposable\_Income', y='Consumption', data=df, s=100, color='blue', label='Actual Data')

plt.plot(df['Disposable\_Income'], df['Predicted\_Consumption'], color='red',

label=f'Regression Line (R²={r\_squared:.2f}, p={p\_value:.4f})')

plt.title('Consumption vs. Disposable Income (2014-2023)', fontsize=14)

plt.xlabel('Disposable Income (INR in CRORES)', fontsize=12)

plt.ylabel('Consumption (INR in CRORES)', fontsize=12)

plt.legend()

plt.grid(True)

# Annotate regression equation and p-value

plt.text(0.05, 0.85,

f'Equation: Consumption = {slope:.6f}\*Income + {intercept:.1f}\n'

f'p-value: {p\_value:.4f} ({"Significant" if p\_value < 0.05 else "Not Significant"})',

transform=plt.gca().transAxes, bbox=dict(facecolor='white', alpha=0.8))

plt.show()

# 4. Print Full Regression Summary

print("\n" + "="\*50)

print("Detailed Regression Summary (statsmodels):")

print("="\*50)

print(model\_sm.summary())

# 5. Key Metrics Table

print("\nKey Metrics:")

metrics\_df = pd.DataFrame({

'Metric': ['Slope (MPC)', 'Intercept', 'R-squared', 'p-value'],

'Value': [slope, intercept, r\_squared, p\_value],

'Interpretation': [

f'Marginal Propensity to Consume: {slope:.6f}',

f'Base consumption when income=0: {intercept:.1f}',

f'Variance explained: {r\_squared:.2%}',

f'{"Significant (p < 0.05)" if p\_value < 0.05 else "Not significant"}'

]

})

print(metrics\_df.to\_string(index=False))