VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Winter Semester 2024 –2025
PMDTS 504L – Regression Analysis and Predictive Modelling
Experiment 2

Study and Implementation of Simple Linear Regression in Python programming language

Objective:

Use Python to perform a simple linear regression analysis on Canada's per capita income data, split the data into training and testing sets, and evaluate the model. The data in .doc file is available in VITOP.

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
import numpy as np

# Step 1: Load the data from CSV file
# File path where the data is stored
#data_path = "/mnt/data/canada_income.csv"
# Read the CSV file into a pandas DataFrame
#data = pd.read_csv(data_path)

filename = "Stock_data.csv"  # Ensure your CSV is named accordingly
data = pd.read_csv(filename)

# Read the CSV file into a pandas DataFrame

# Display the first few rows to confirm the data is loaded correctly
print("Data Preview:")
print(data.head())
```

```
# Step 2: Prepare the data for the regression model
# Drop any rows with missing values (if any)
data = data.dropna()

# Convert 'year' column to integers (ensures compatibility with sklearn)
data['year'] = data['year'].astype(int)

# Define predictor (X) and target (y) variables
X = data['year'].values.reshape(-1, 1)  # Reshape for sklearn
# Predictor: Years (reshaped into 2D array)
y = data['per capita income (US$)'].values
# Target: Per capita income
```

```
# Step 3: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 4: Create and fit the Linear Regression model
# Initialize the Linear Regression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Step 5: Predict the per capita income for the year 2025
# Define the input year for prediction (2025)
year_2025 = np.array([[2025]])

# Use the trained model to make a prediction
income_2025 = model.predict(year_2025)

# Display the predicted per capita income for 2025
print(f"Predicted per capita income for 2025: ${income_2025[0]:,.2f}")
```

```
# Step 6: Evaluate the model performance on the testing set
# Predict the per capita income for the testing set
y_pred = model.predict(X_test)

# Calculate the Mean Squared Error (MSE)
mse_test = mean_squared_error(y_test, y_pred)

# Calculate the R-squared value on the testing set
r2_test = model.score(X_test, y_test)

# Display evaluation metrics for the testing set
print(f"Testing Set Mean Squared Error: {mse_test:.2f}")
print(f"Testing Set R-squared: {r2_test:.2f}")
```

```
# Step 7: Visualize the training data, testing data, and the regression model
plt.figure(figsize=(10, 6))  # Set the figure size

# Scatter plot of the training data
plt.scatter(X_train, y_train, color='blue', label='Training Data', alpha=0.7)

# Scatter plot of the testing data
plt.scatter(X_test, y_test, color='orange', label='Testing Data', alpha=0.7)

# Plot the regression line
plt.plot(X, model.predict(X), color='red', label='Regression Line')

# Highlight the prediction for 2025
plt.scatter(year_2025, income_2025, color='green', label='2025 Prediction', zorder=5, marker='o', s=100)

# Add labels, title, and legend
plt.xlabel('Year')
plt.ylabel('Per Capita Income (US$)')
plt.title('Canada Per Capita Income Prediction')
plt.legend()
plt.grid(True)

# Show the plot
plt.show()
```

```
# Step 8: Evaluate the model performance on the training set (optional)
# Predict the per capita income for the training set
y_train_pred = model.predict(X_train)

# Calculate the Mean Squared Error (MSE) for the training set
mse_train = mean_squared_error(y_train, y_train_pred)

# Calculate the R-squared value on the training set
r2_train = model.score(X_train, y_train)

# Display evaluation metrics for the training set
print(f"Training Set Mean Squared Error: {mse_train:.2f}")
print(f"Training Set R-squared: {r2_train:.2f}")
```

```
#Display the regression coefficients
print("Regression Coefficient (Slope):", model.coef_[0])
print("Regression Intercept:", model.intercept_)
```

```
Regression Coefficient (Slope): 815.1425130089498
Regression Intercept: -1605560.1987964255
```

The provided code performs the following tasks:

1. **Data Loading**: Reads a dataset containing Canada's per capita income over the years from a CSV file.

2. **Data Preparation**: Cleans the data, converts the year column to integers, and splits it into predictors (X) and targets (y).
3. **Data Splitting**: Divides the dataset into training (80%) and testing (20%) sets using train_test_split.
4. **Model Training**: Fits a simple linear regression model to the training data to predict per capita income based on the year.
5. **Prediction**: Uses the trained model to predict Canada's per capita income for the year 2025.
6. **Evaluation**: Calculates the Mean Squared Error (MSE) and R-squared values for both training and testing datasets to evaluate model performance.
7. **Visualization**: Plots the training data, testing data, regression line, and the predicted value for 2025.
8. **Output**: Prints the predicted income for 2025 and evaluation metrics for the model.