Assignment No 6

Aim: Assignment on Regression technique. Download temperature data from below link. https://www.kaggle.com/venky73/temperaturesof-india?select=temperatures. csv This data consists of temperatures of INDIA averaging the temperatures of all places month wise. Temperatures values are recorded in CELSIUS a) Apply Linear Regression using suitable library function and predict the Month-wise temperature. b) Assess the performance of regression models using MSE, MAE and R-Square metrics c) Visualize simple regression model.

Dataset Overview

Dataset Source: Kaggle - Temperatures of India Dataset
Link: https://www.kaggle.com/venky73/temperaturesof-india

This dataset consists of average monthly temperatures of India. Temperature values are recorded in degrees Celsius, and represent nationwide averages.

Basic Concepts & Theory

◇ Linear Regression

Linear Regression is a supervised learning algorithm used for predicting a quantitative response. It models the relationship between a dependent variable (target) and one or more independent variables (features) using a linear equation.

For Simple Linear Regression:
$y = \beta0 + \beta1 * x + \varepsilon$

Where:
- y is the target variable (temperature)
- x is the independent variable (month)
- β0 is the intercept
- β1 is the slope of the line
- ε is the error term

Methodology

The following steps were used to complete the assignment:
1. Load the dataset using pandas.
2. Explore and preprocess the data if needed.
3. Apply Linear Regression using scikit-learn's `LinearRegression()` model.
4. Use the fitted model to predict monthly temperatures.

5. Evaluate model performance using the following metrics:
   - Mean Squared Error (MSE)
   - Mean Absolute Error (MAE)
   - R-Squared Score (R^2)
6. Visualize the regression line using matplotlib.

Performance Evaluation Metrics

- Mean Squared Error (MSE): Measures average squared difference between actual and predicted values. Lower is better.
- Mean Absolute Error (MAE): Measures average absolute error. Lower is better.
- R-Squared (R^2): Indicates goodness of fit. Ranges from 0 to 1. Closer to 1 means better fit.

Output:

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

In [5]:

```
# Load dataset
df = pd.read_csv("temperatures.csv")

# Display first few rows
print(df.head())

# Check for missing values
print(df.isnull().sum())

# Summary statistics
print(df.describe())
```

```
   YEAR    JAN    FEB    MAR    APR    MAY    JUN    JUL    AUG    SEP    OCT  \
0  1901  22.40  24.14  29.07  31.91  33.41  33.18  31.21  30.39  30.47  29.97
1  1902  24.93  26.58  29.77  31.78  33.73  32.91  30.92  30.73  29.80  29.12
2  1903  23.44  25.03  27.83  31.39  32.91  33.00  31.34  29.98  29.85  29.04
3  1904  22.50  24.73  28.21  32.02  32.64  32.07  30.36  30.09  30.04  29.20
4  1905  22.00  22.83  26.68  30.01  33.32  33.25  31.44  30.68  30.12  30.67

    NOV    DEC  ANNUAL  JAN-FEB  MAR-MAY  JUN-SEP  OCT-DEC
0  27.31  24.49   28.96    23.27    31.46    31.27    27.25
1  26.31  24.04   29.22    25.75    31.76    31.09    26.49
2  26.08  23.65   28.47    24.24    30.71    30.92    26.26
3  26.36  23.63   28.49    23.62    30.95    30.66    26.40
4  27.52  23.82   28.30    22.25    30.00    31.33    26.57
YEAR        0
JAN         0
FEB         0
MAR         0
APR         0
MAY         0
JUN         0
JUL         0
AUG         0
SEP         0
OCT         0
NOV         0
DEC         0
ANNUAL      0
JAN-FEB     0
MAR-MAY     0
JUN-SEP     0
OCT-DEC     0
dtype: int64
              YEAR         JAN         FEB         MAR         APR  \
count   117.000000  117.000000  117.000000  117.000000  117.000000
mean   1959.000000   23.687436   25.597863   29.085983   31.975812
std      33.919021    0.834588    1.150757    1.068451    0.889478
min    1901.000000   22.000000   22.830000   26.680000   30.010000
25%    1930.000000   23.100000   24.780000   28.370000   31.460000
50%    1959.000000   23.680000   25.480000   29.040000   31.950000
75%    1988.000000   24.180000   26.310000   29.610000   32.420000
max    2017.000000   26.940000   29.720000   32.620000   35.380000

              MAY         JUN         JUL         AUG         SEP         OCT  \
count  117.000000  117.000000  117.000000  117.000000  117.000000  117.000000
mean    33.565299   32.774274   31.035897   30.507692   30.486752   29.766581
std      0.724905    0.633132    0.468818    0.476312    0.544295    0.705492
min     31.930000   31.100000   29.760000   29.310000   29.070000   27.900000
25%     33.110000   32.340000   30.740000   30.180000   30.120000   29.380000
50%     33.510000   32.730000   31.000000   30.540000   30.520000   29.780000
75%     34.030000   33.180000   31.330000   30.760000   30.810000   30.170000
max     35.840000   34.480000   32.760000   31.840000   32.220000   32.290000

              NOV         DEC      ANNUAL     JAN-FEB     MAR-MAY     JUN-SEP  \
count  117.000000  117.000000  117.000000  117.000000  117.000000  117.000000
mean    27.285470   24.608291   29.181368   24.629573   31.517607   31.198205
std      0.714518    0.782644    0.555555    0.911239    0.740585    0.420508
min     25.700000   23.020000   28.110000   22.250000   29.920000   30.240000
25%     26.790000   24.040000   28.760000   24.110000   31.040000   30.920000
50%     27.300000   24.660000   29.090000   24.530000   31.470000   31.190000
75%     27.720000   25.110000   29.470000   25.150000   31.890000   31.400000
max     30.110000   28.010000   31.630000   28.330000   34.570000   32.410000

          OCT-DEC
count  117.000000
mean    27.208120
std      0.672003
min     25.740000
25%     26.700000
50%     27.210000
75%     27.610000
max     30.030000
```

In [7]:
```
print("Columns in dataset:", df.columns)
```

```
Columns in dataset: Index(['YEAR', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP',
       'OCT', 'NOV', 'DEC', 'ANNUAL', 'JAN-FEB', 'MAR-MAY', 'JUN-SEP',
       'OCT-DEC'],
      dtype='object')
```

In [8]:
```
# Define features (independent variable) and target (dependent variable)
X = df[['YEAR']]  # Feature
y = df.drop(columns=['YEAR', 'ANNUAL', 'JAN-FEB', 'MAR-MAY', 'JUN-SEP', 'OCT-DEC'])  # Target variables (all months)
```

In [9]:
```
# Split 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)
```

In [10]:
```
# Initialize and train Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

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

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

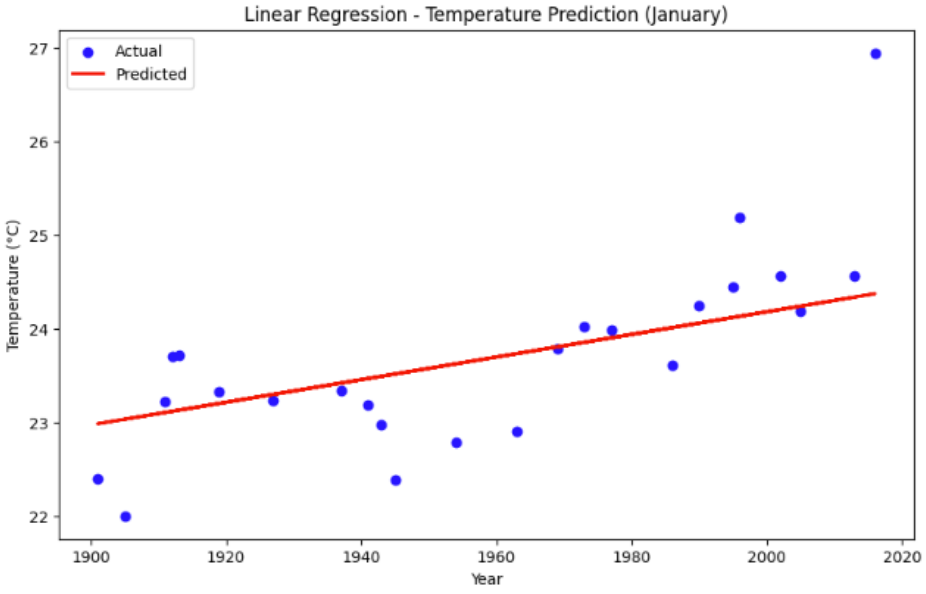
In [11]:
```
# Make predictions
y_pred = model.predict(X_test)
```

In [12]:
```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse:.4f}")
print(f"Mean Absolute Error (MAE): {mae:.4f}")
print(f"R-Square Score: {r2:.4f}")
```

```
Mean Squared Error (MSE): 0.4944
Mean Absolute Error (MAE): 0.5028
R-Square Score: 0.2906
```

In [13]:
```
# Visualizing the Linear Regression for one month (e.g., January)
plt.figure(figsize=(10, 6))
plt.scatter(X_test, y_test['JAN'], color='blue', label='Actual')
plt.plot(X_test, y_pred[:, 0], color='red', linewidth=2, label='Predicted')  # JAN is the first column in y_pred
plt.xlabel("Year")
plt.ylabel("Temperature (°C)")
plt.title("Linear Regression - Temperature Prediction (January)")
plt.legend()
plt.show()
```



Conclusion

In this assignment, we successfully performed temperature prediction using Linear Regression on the Indian temperature dataset. We evaluated the model using MSE, MAE, and R-squared, and visualized the regression line.