|  |  |
| --- | --- |
| **Name: Jal Bafana** | **Roll no: K005** |
| **Btech. Cyber Security Sem-4** | **Batch: K1** |
| **Date of Experiment: 11.02.2025** | **Date of Submission: 11.02.2025** |

**Lab-5**

from google.colab import drive

drive.mount('/content/drive')

**saving the process**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

**importing the libraries**

file\_path = '/content/drive/MyDrive/Summary-Weather.csv'

**saving the path for downloaded csv**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error, r2\_score

**importing the libraries**

df = pd.read\_csv(file\_path)

**defining the dataframe**

# Check the column names in your dataset

print(df.columns)

**the column names in dataset**

X = data.drop(['Date', 'STA'], axis=1)  # Drop 'Date' and 'STA' (station column)

y = data['MeanTemp']  # Target variable (change this as per your choice)

**dropping date and sta as it contains non-numeric values**

X['PoorWeather'] = X['PoorWeather'].apply(lambda x: 1 if x == 'Yes' else 0)

**Its like if there is a yes in column the 1 and if there is a no in a column then 0**

X = X.apply(pd.to\_numeric, errors='coerce')

X = X.fillna(X.mean())

from sklearn.impute import SimpleImputer

# Initialize the SimpleImputer to fill missing values with the column mean

imputer = SimpleImputer(strategy='mean')

# Apply the imputer to fill missing values in the features (X)

X = imputer.fit\_transform(X)

# Check if NaN values still exist after transformation

print(np.isnan(X).sum())

# If NaN values exist in the target variable (y), handle them too

y = np.array(y)  # Convert y to numpy array if it's not already

y = np.nan\_to\_num(y, nan=np.nanmean(y))  # Replace NaN values in target with the mean

**replacing the NaN values**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = LinearRegression()

model.fit(X\_train, y\_train)

# Predict the target values for the test set

y\_pred = model.predict(X\_test)

# Evaluate the model's performance

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

# Print the metrics

print(f"Mean Squared Error: {mse}")

print(f"R^2 Score: {r2}")

# Plot Actual vs Predicted values

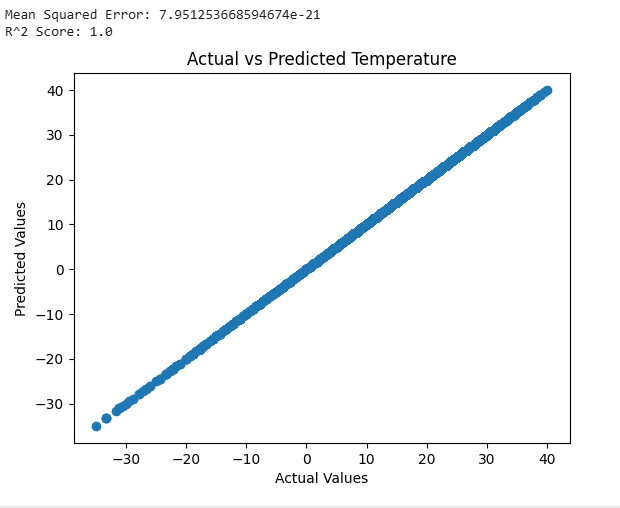
plt.scatter(y\_test, y\_pred)

plt.xlabel('Actual Values')

plt.ylabel('Predicted Values')

plt.title('Actual vs Predicted Temperature')

plt.show()

****