untitled4

November 2, 2023

[1]:
```
import pandas as pd
```

[2]:
```
df=pd.read_csv("cocomo 81.csv")
```

[3]:
```
df.head()
```

[3]:
```
   Rely  Data  Cplx  Time  Stor  Virt  Turn  Acap  Aexp  Pcap  Vexp  Lexp  \
0  0.88  1.16  0.70   1.0  1.06  1.15  1.07  1.19  1.13  1.17   1.1  1.00
1  0.88  1.16  0.85   1.0  1.06  1.00  1.07  1.00  0.91  1.00   0.9  0.95
2  1.00  1.16  0.85   1.0  1.00  0.87  0.94  0.86  0.82  0.86   0.9  0.95
3  0.75  1.16  0.70   1.0  1.00  0.87  1.00  1.19  0.91  1.42   1.0  0.95
4  0.88  0.94  1.00   1.0  1.00  0.87  1.00  1.00  1.00  0.86   0.9  0.95

   Modp  Tool  Sced   Size  Effort1
0  1.24  1.10  1.04  113.0   2040.0
1  1.10  1.00  1.00  293.0   1600.0
2  0.91  0.91  1.00  132.0    243.0
3  1.24  1.00  1.04   60.0    240.0
4  1.24  1.00  1.00   16.0     33.0
```

[4]:
```
X = df.drop(columns=['Effort1'])
Y = df['Effort1']
```

[5]:
```
from sklearn.model_selection import train_test_split
# Split the data into features (X) and the target variable (Y)
# Assuming you've already split the data as shown in the previous response
# 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)

# Parameters:
# - X: Features
# - Y: Target variable
# - test_size: The proportion of the dataset to include in the test split (e.g.
 ↪, 0.2 for a 80% train / 20% test split)
# - random_state: Optional seed for reproducibility
# Now you have:
# - X_train: Features for training
# - X_test: Features for testing
# - Y_train: Target variable for training
# - Y_test: Target variable for testing
```

[6]:

```
from sklearn.tree import DecisionTreeRegressor
```

[7]:

```
from sklearn.model_selection import train_test_split
```

[12]:

```
regressor = DecisionTreeRegressor(random_state=7)
```

[9]:

```
regressor.fit(X_train, Y_train)
```

[9]:

```
DecisionTreeRegressor(random_state=42)
```

[10]:

```
Y_pred = regressor.predict(X_test)
```

[11]:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Calculate regression metrics
mae = mean_absolute_error(Y_test, Y_pred)
mse = mean_squared_error(Y_test, Y_pred)
r2 = r2_score(Y_test, Y_pred)

print("Mean Absolute Error: ", mae)
print("Mean Squared Error: ", mse)
print("R-squared (R^2): ", r2)
```

```
Mean Absolute Error:  304.7692307692308
Mean Squared Error:  351599.53846153844
R-squared (R^2):  -0.13777512821729432
```

[ ]:

```

```