```python
#multilinear model
import pandas as pd
import numpy as np
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
data = pd.read_csv(r"C:\Users\arpit\Downloads\Real estate - Real estate (1).csv")
df1 = pd.DataFrame(data)
print(df1)
df2 = df1.drop(["Unnamed: 8"],axis = 1)
df2.isnull()
df3 = df2.dropna()
print(df3)
print(df3.duplicated())
df3 = df3.drop_duplicates()
print(df3)
df4=df3.duplicated(subset=['Y house price of unit area'])
df4 = df3.drop_duplicates(subset=['Y house price of unit area'],keep='last')
print(df4)
plt.scatter(df4['X5 latitude'],df4['Y house price of unit area'],color='red')
plt.scatter(df4['X6 longitude'],df4['Y house price of unit area'],color='g')
plt.show()
regr = linear_model.LinearRegression()
x=df4[['X5 latitude','X6 longitude']]
y = np.asanyarray(df4['Y house price of unit area'])
print(x)
print(y)
Y=regr.fit(x,y)
print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)
b = regr.predict([[15,9]])
print("Y house price of unit area: {}".format(b))
```
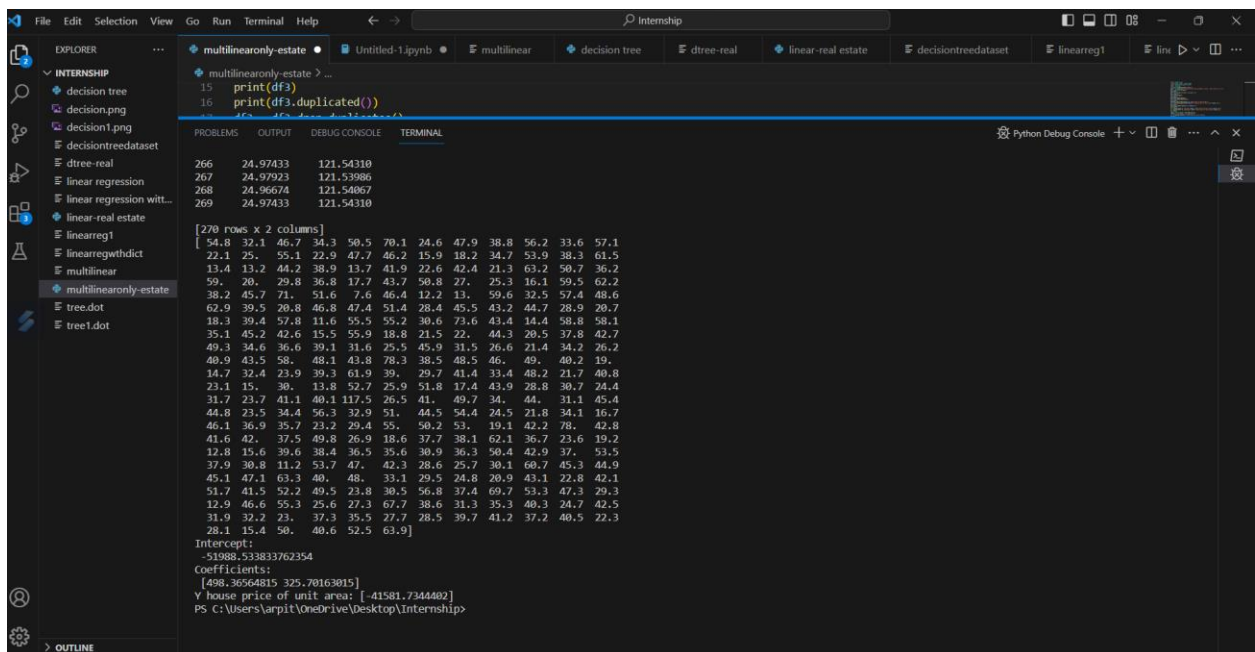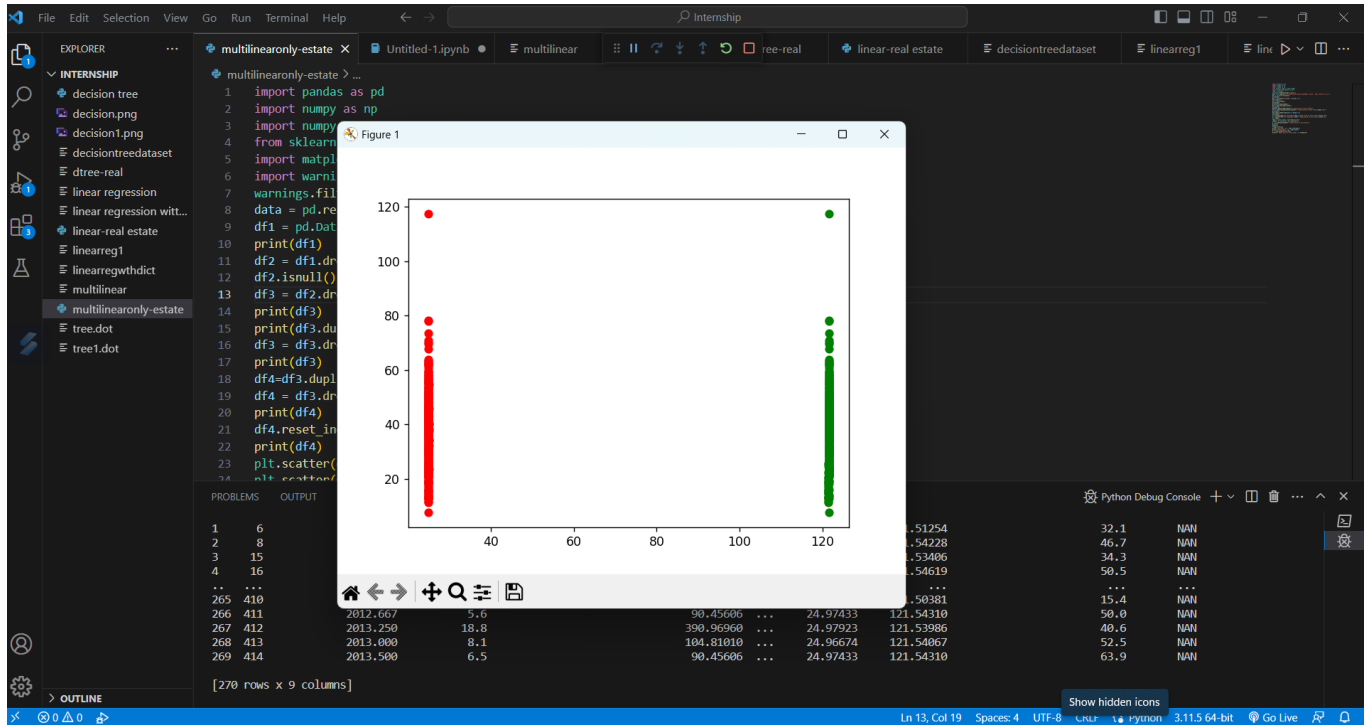
Screenshot:

Linear model

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error
from math import sqrt

# Load the dataset
data = pd.read_csv(r"C:\Users\arpit\Downloads\Real estate - Real estate (1).csv")
df1 = pd.DataFrame(data)

# Drop the "Unnamed: 8" column
df1 = df1.drop(["Unnamed: 8"], axis=1)

# Remove rows with missing values
df2 = df1.dropna()

# Remove duplicate rows
df3 = df2.drop_duplicates()

# Reset the index
df3.reset_index(inplace=True, drop=True)

# Plot the data
plt.scatter(df3['X2 house age'], df3['Y house price of unit area'], color='red')
plt.xlabel('X2 house age')
plt.ylabel('Y house price of unit area')
plt.show()
```

```python
# Define the features (X) and target (y)
X = df3[['X2 house age']]
y = df3['Y house price of unit area']

# 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)

# Create a linear regression model
regr = LinearRegression()

# Train the model
regr.fit(x_train, y_train)

# Make predictions on the test set
y_pred = regr.predict(x_test)

# Plot the regression line
plt.scatter(x_test, y_test, color='red')
plt.plot(x_test, y_pred, color='blue', linewidth=3)
plt.xlabel('X2 house age')
plt.ylabel('Y house price of unit area')
plt.show()

# Input a value for "X2 house age" and make a prediction
d = float(input('Enter the value of X2 house age: '))
predicted_price = regr.predict([[d]])[0]
print(f'Predicted Y house price of unit area: {predicted_price:.2f}')

# Calculate and print evaluation metrics
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
```
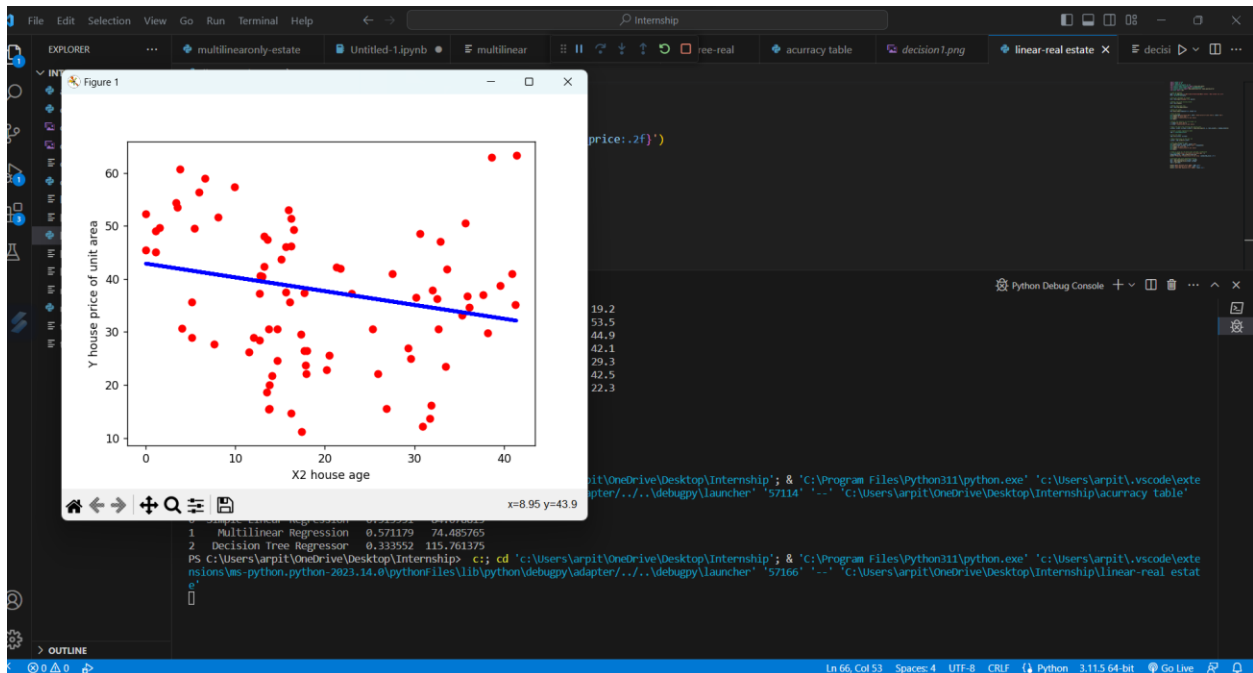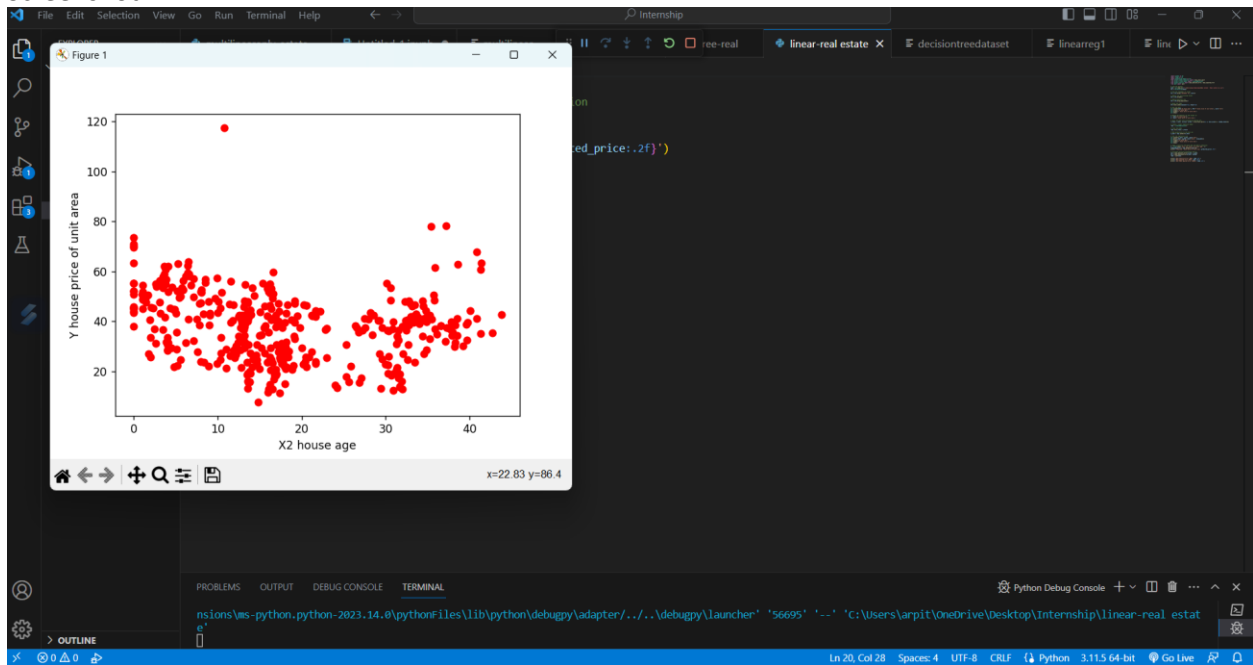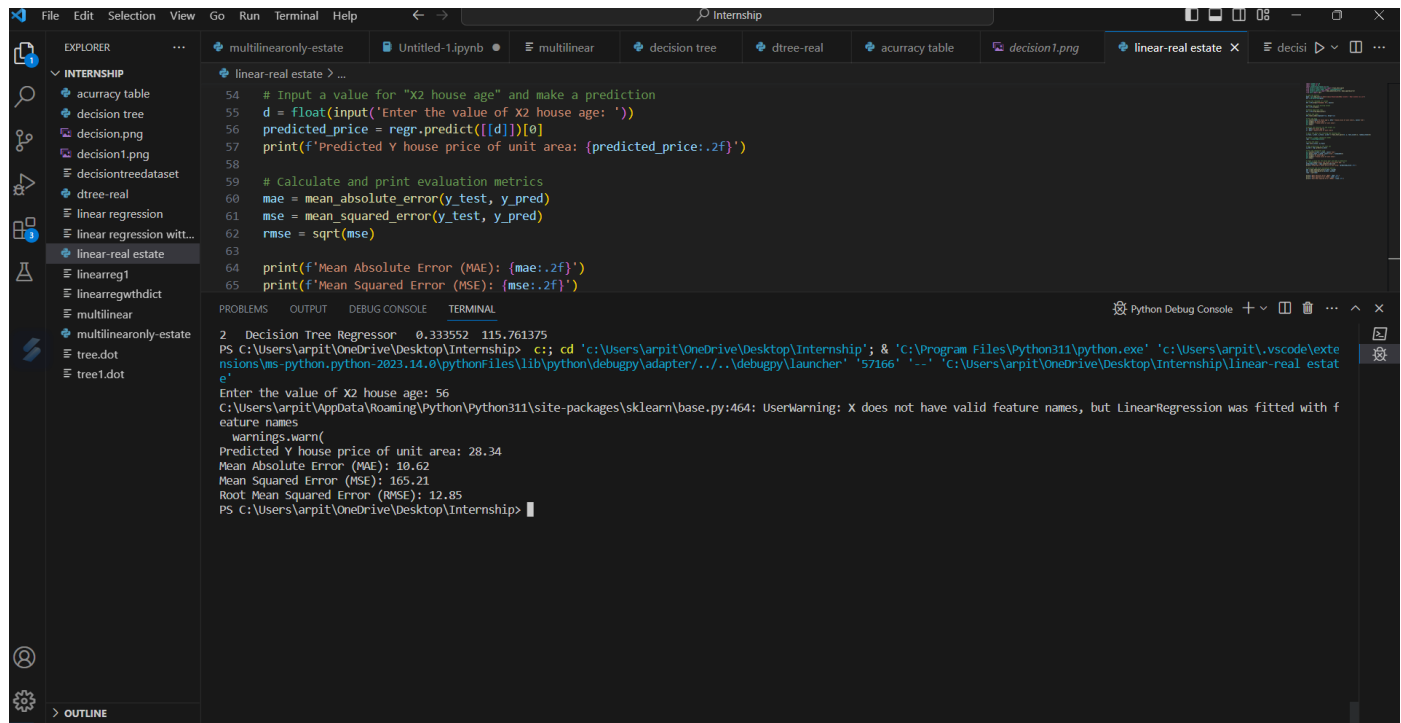
```python
mse = mean_squared_error(y_test, y_pred)
rmse = sqrt(mse)

print(f'Mean Absolute Error (MAE): {mae:.2f}')
print(f'Mean Squared Error (MSE): {mse:.2f}')
print(f'Root Mean Squared Error (RMSE): {rmse:.2f}')
```

Screenshot:

```python
54    # Input a value for "X2 house age" and make a prediction
55    d = float(input('Enter the value of X2 house age: '))
56    predicted_price = regr.predict([[d]])[0]
57    print(f'Predicted Y house price of unit area: {predicted_price:.2f}')
58
59    # Calculate and print evaluation metrics
60    mae = mean_absolute_error(y_test, y_pred)
61    mse = mean_squared_error(y_test, y_pred)
62    rmse = sqrt(mse)
63
64    print(f'Mean Absolute Error (MAE): {mae:.2f}')
65    print(f'Mean Squared Error (MSE): {mse:.2f}')
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
2    Decision Tree Regressor   0.333552  115.761375
PS C:\Users\arpit\OneDrive\Desktop\Internship>  c:; cd 'c:\Users\arpit\OneDrive\Desktop\Internship'; & 'C:\Program Files\Python311\python.exe' 'c:\Users\arpit\.vscode\exte
nsions\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '57166' '--' 'C:\Users\arpit\OneDrive\Desktop\Internship\linear-real estat
e'
Enter the value of X2 house age: 56
C:\Users\arpit\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but LinearRegression was fitted with f
eature names
  warnings.warn(
Predicted Y house price of unit area: 28.34
Mean Absolute Error (MAE): 10.62
Mean Squared Error (MSE): 165.21
Root Mean Squared Error (RMSE): 12.85
PS C:\Users\arpit\OneDrive\Desktop\Internship>
```

```python
#decision tree
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
from sklearn import tree

# Load the data
data = pd.read_csv('C:/Users/HP/Downloads/Real estate - Real estate.csv')

# Drop unnecessary columns and remove rows with missing values
df = data.drop(["No", "X1 transaction date", "X5 latitude", "X6 longitude"],
axis=1).dropna()

# Remove duplicate rows
df = df.drop_duplicates()

# Split the data into input features (X) and target variable (Y)
X = df[['X3 distance to the nearest MRT station']]
Y = df['Y house price of unit area']

# 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=0)

# Create and fit the DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state=0)
regressor.fit(X_train, Y_train)

# Plot the decision tree (optional)
plt.figure(figsize=(10, 6))
tree.plot_tree(regressor, filled=True, feature_names=['X3 distance to the nearest MRT
station'])
plt.show()

# Export the decision tree to a DOT file (optional)
from sklearn.tree import export_graphviz
export_graphviz(regressor, out_file='tree.dot', feature_names=['X3 distance to the nearest
MRT station'])

# Make predictions
y_pred = regressor.predict([[390]])
print("Predicted Y house price of unit area:", y_pred[0])
```

Screenshot

OUTPUT:

ACCURACY TABLE:

```python
1   import pandas as pd
2   import numpy as np
3   from sklearn import linear_model
4   from sklearn.tree import DecisionTreeRegressor
5   from sklearn.model_selection import train_test_split
6   from sklearn.metrics import r2_score, mean_squared_error
7
8   # Load the data
9   data = pd.read_csv(r"C:\Users\arpit\Downloads\Real estate - Real estate.csv")
10
11  # Drop unnecessary columns and remove rows with missing values
12  df = data.drop(["No", "X1 transaction date", "X5 latitude", "X6 longitude"], axis=1).dropna()
13
14  # Remove duplicate rows
15  df = df.drop_duplicates()
16
17  # Split the data into input features (X) and target variable (Y)
18  X_linear = df[['X3 distance to the nearest MRT station']]
19  X_multilinear = df[['X3 distance to the nearest MRT station', 'X4 number of convenience stores']]
20  X_tree = df[['X3 distance to the nearest MRT station']]
21  Y = df['Y house price of unit area']
22
23  # Split the data into training and testing sets
24  X_linear_train, X_linear_test, Y_train, Y_test = train_test_split(X_linear, Y, test_size=0.2, random_state=0)
25  X_multilinear_train, X_multilinear_test, _, _ = train_test_split(X_multilinear, Y, test_size=0.2, random_state=0)
26  X_tree_train, X_tree_test, _, _ = train_test_split(X_tree, Y, test_size=0.2, random_state=0)
27
28  # Create and fit the Linear Regression model
29  regr_linear = linear_model.LinearRegression()
30  regr_linear.fit(X_linear_train, Y_train)
31
32  # Make predictions using Linear Regression
33  y_pred_linear = regr_linear.predict(X_linear_test)
34
35  # Calculate R-squared and MSE for Linear Regression
36  r2_linear = r2_score(Y_test, y_pred_linear)
37  mse_linear = mean_squared_error(Y_test, y_pred_linear)
```

```python
38
39    # Create and fit the Multilinear Regression model
40    regr_multilinear = linear_model.LinearRegression()
41    regr_multilinear.fit(X_multilinear_train, Y_train)
42
43    # Make predictions using Multilinear Regression
44    y_pred_multilinear = regr_multilinear.predict(X_multilinear_test)
45
46    # Calculate R-squared and MSE for Multilinear Regression
47    r2_multilinear = r2_score(Y_test, y_pred_multilinear)
48    mse_multilinear = mean_squared_error(Y_test, y_pred_multilinear)
49
50    # Create and fit the Decision Tree Regressor model
51    regressor_tree = DecisionTreeRegressor(random_state=0)
52    regressor_tree.fit(X_tree_train, Y_train)
53
54    # Make predictions using Decision Tree Regressor
55    y_pred_tree = regressor_tree.predict(X_tree_test)
56
57    # Calculate R-squared and MSE for Decision Tree Regressor
58    r2_tree = r2_score(Y_test, y_pred_tree)
59    mse_tree = mean_squared_error(Y_test, y_pred_tree)
60
61    # Create an accuracy table
62    accuracy_table = pd.DataFrame({
63        'Model': ['Simple Linear Regression', 'Multilinear Regression', 'Decision Tree Regressor'],
64        'R-squared': [r2_linear, r2_multilinear, r2_tree],
65        'MSE': [mse_linear, mse_multilinear, mse_tree]
66    })
67
68    print(accuracy_table)
```

OUTPUT:

Based on R-squared values, the Multilinear Regression model (0.571179) has the highest R-squared, which suggests it explains more variance in the data compared to the other models. Therefore, it seems to be the most accurate among the three models in terms of explaining the variation in the dependent variable.

Based on MSE values, the Multilinear Regression model also has the lowest MSE (74.485765), which indicates it has the smallest error in predicting the target variable compared to the other models. So, it is the most accurate in terms of minimizing prediction error.

In conclusion, based on both R-squared and MSE, the Multilinear Regression model appears to be the most accurate among the three models provided