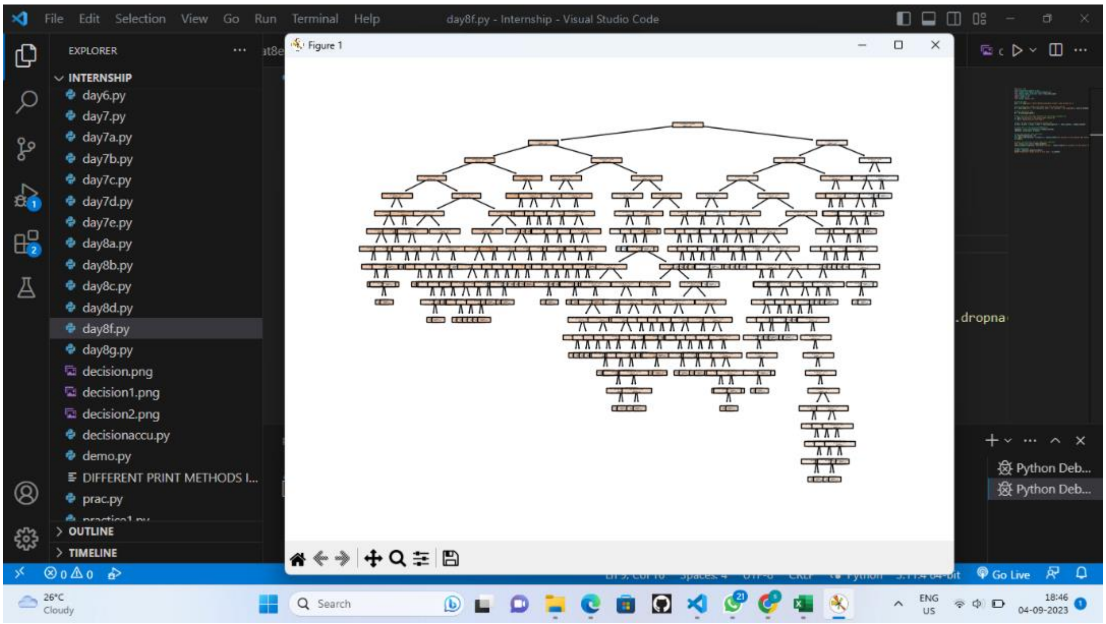
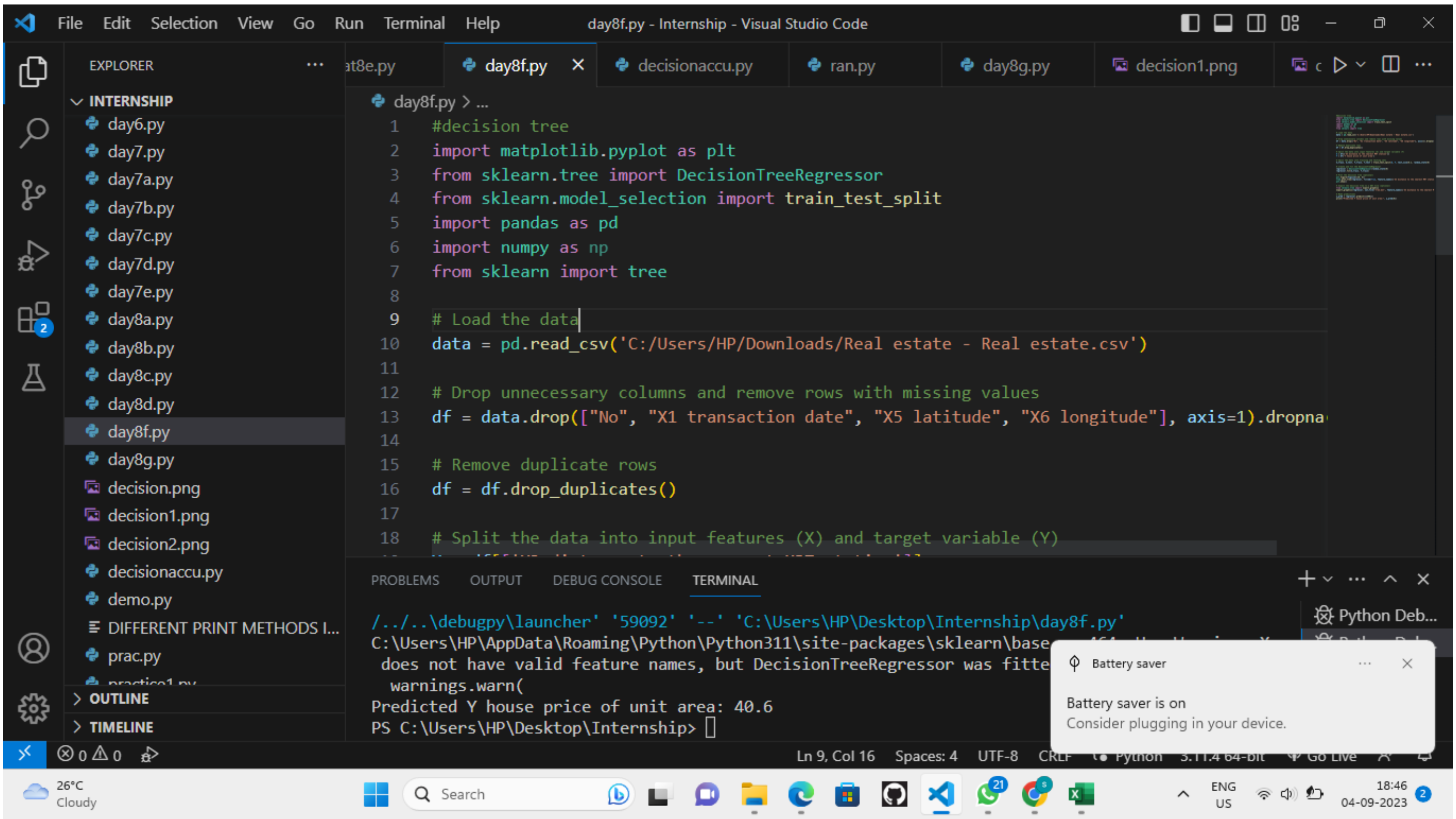
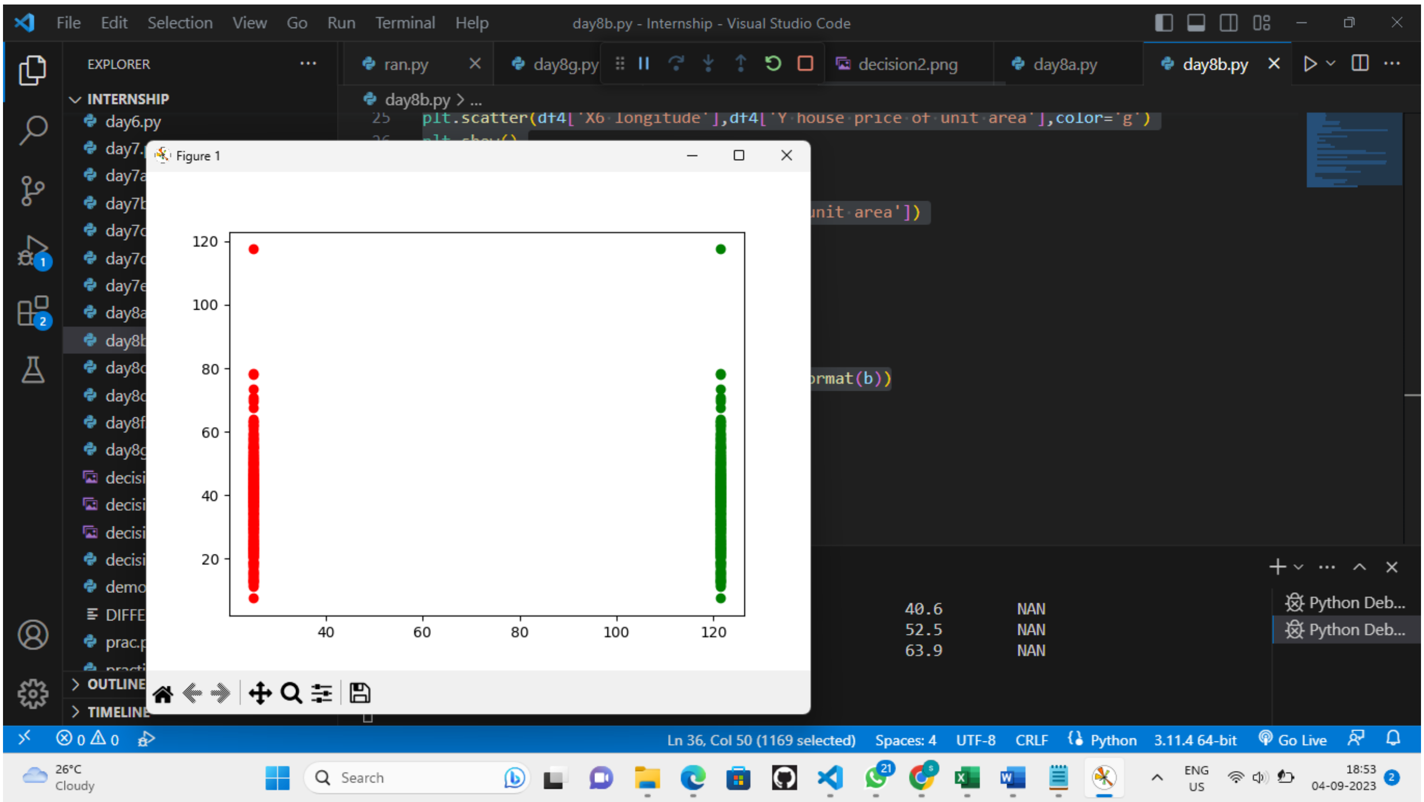
```
#decision tree
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
from sklearn import tree

# 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])
```

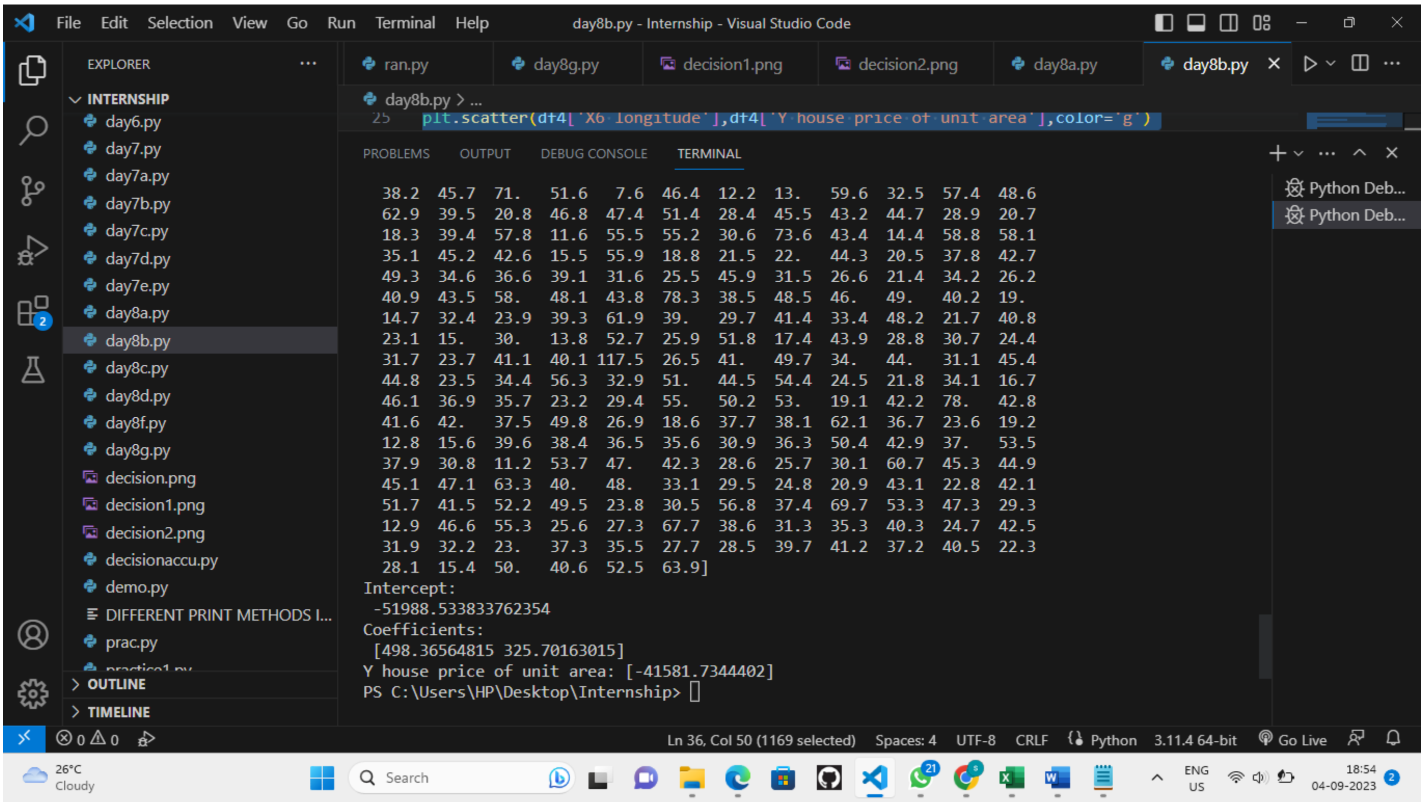
screenshot:


File Edit Selection View Go Run Terminal Help
day8f.py - Internship - Visual Studio Code
EXPLORER
INTERNSHIP
day6.py
day7.py
day7a.py
day7b.py
day7c.py
day7d.py
day7e.py
day8a.py
day8b.py
day8c.py
day8d.py
day8f.py
day8g.py
decision.png
decision1.png
decision2.png
decisionaccu.py
demo.py
DIFFERENT PRINT METHODS I...
prac.py
OUTLINE
TIMELINE
Figure 1
Python Deb...
Python Deb...
Go Live
26°C
Cloudy
Search
ENG
US
18:46
04-09-2023


File Edit Selection View Go Run Terminal Help
day8f.py - Internship - Visual Studio Code
EXPLORER
at8e.py
day8f.py
decisionaccu.py
ran.py
day8g.py
decision1.png
INTERNSHIP
day6.py
day7.py
day7a.py
day7b.py
day7c.py
day7d.py
day7e.py
day8a.py
day8b.py
day8c.py
day8d.py
day8f.py
day8g.py
decision.png
decision1.png
decision2.png
decisionaccu.py
demo.py
DIFFERENT PRINT METHODS I...
prac.py
OUTLINE
TIMELINE
day8f.py > ...
#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)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
/../..\debugpy\launcher' '59092' '--' 'C:\Users\HP\Desktop\Internship\day8f.py'
C:\Users\HP\AppData\Roaming\Python\Python311\site-packages\sklearn\base
does not have valid feature names, but DecisionTreeRegressor was fitte
warnings.warn(
Predicted Y house price of unit area: 40.6
PS C:\Users\HP\Desktop\Internship>
Python Deb...
Battery saver
Battery saver is on
Consider plugging in your device.
Ln 9, Col 16 Spaces: 4 UTF-8 CRLF Python 3.11.4 64-bit Go Live
26°C
Cloudy
Search
ENG
US
18:46
04-09-2023

```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/HP\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)
df4.reset_index(inplace=True,drop=True)
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))
```

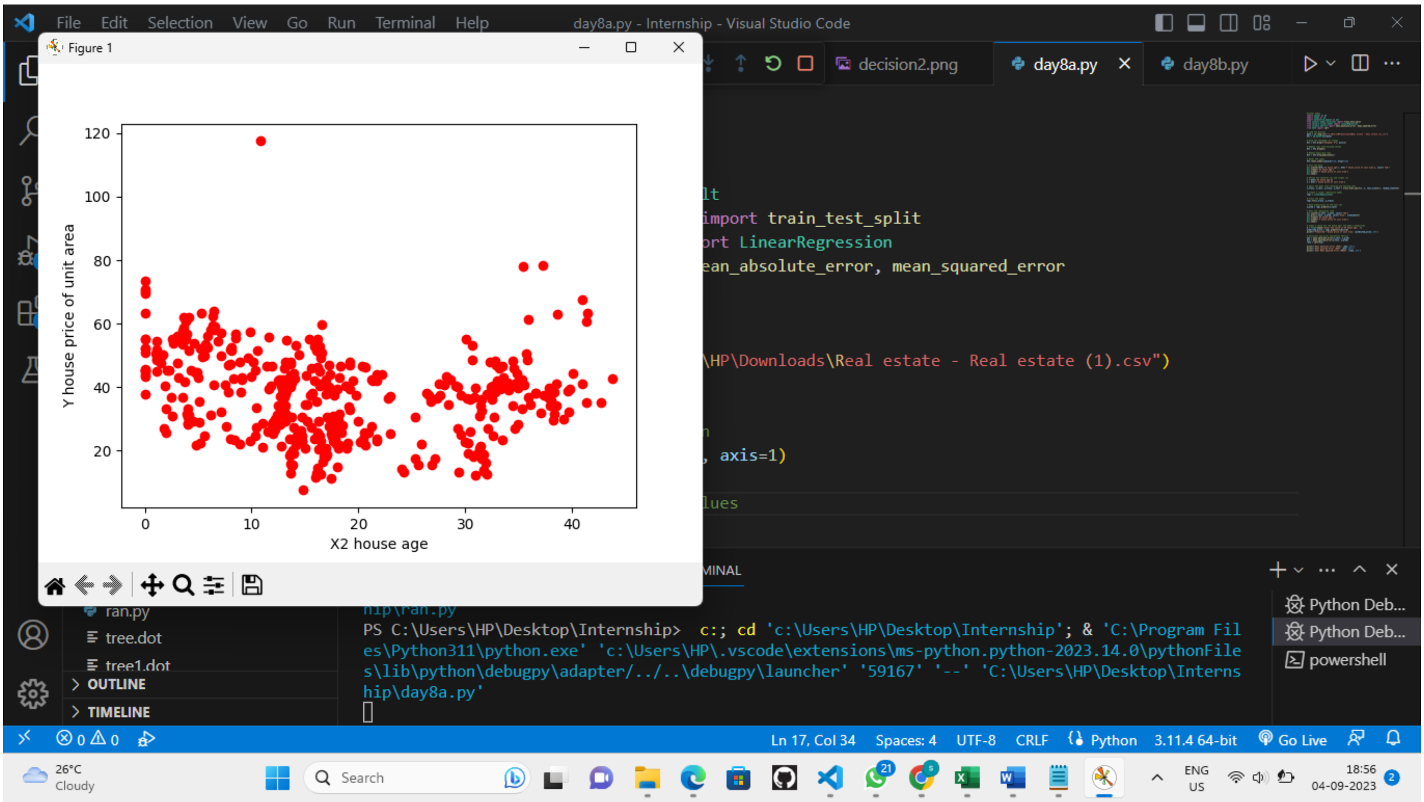




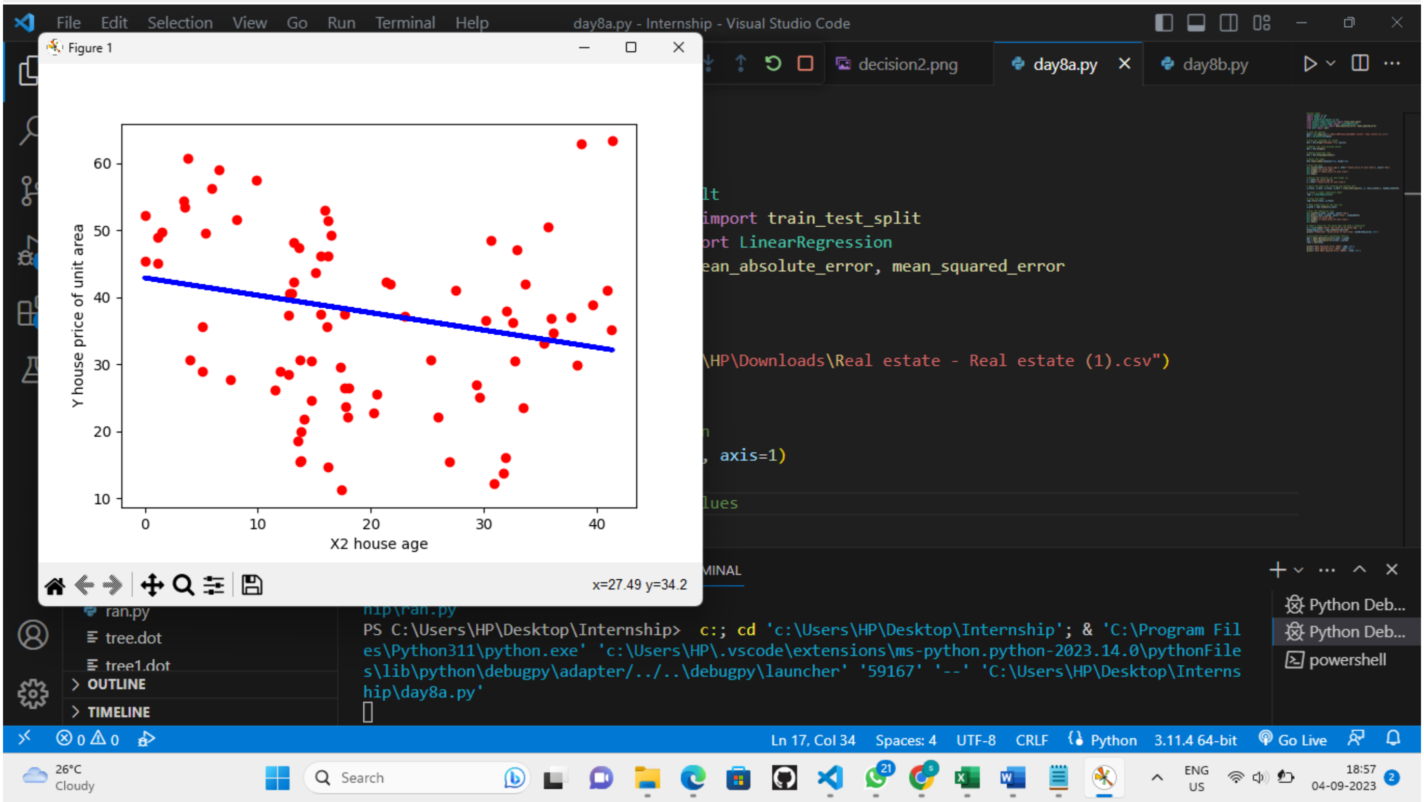

```
#linear model
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error
from math import sqrt

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

```python
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()

# 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)
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}')
```

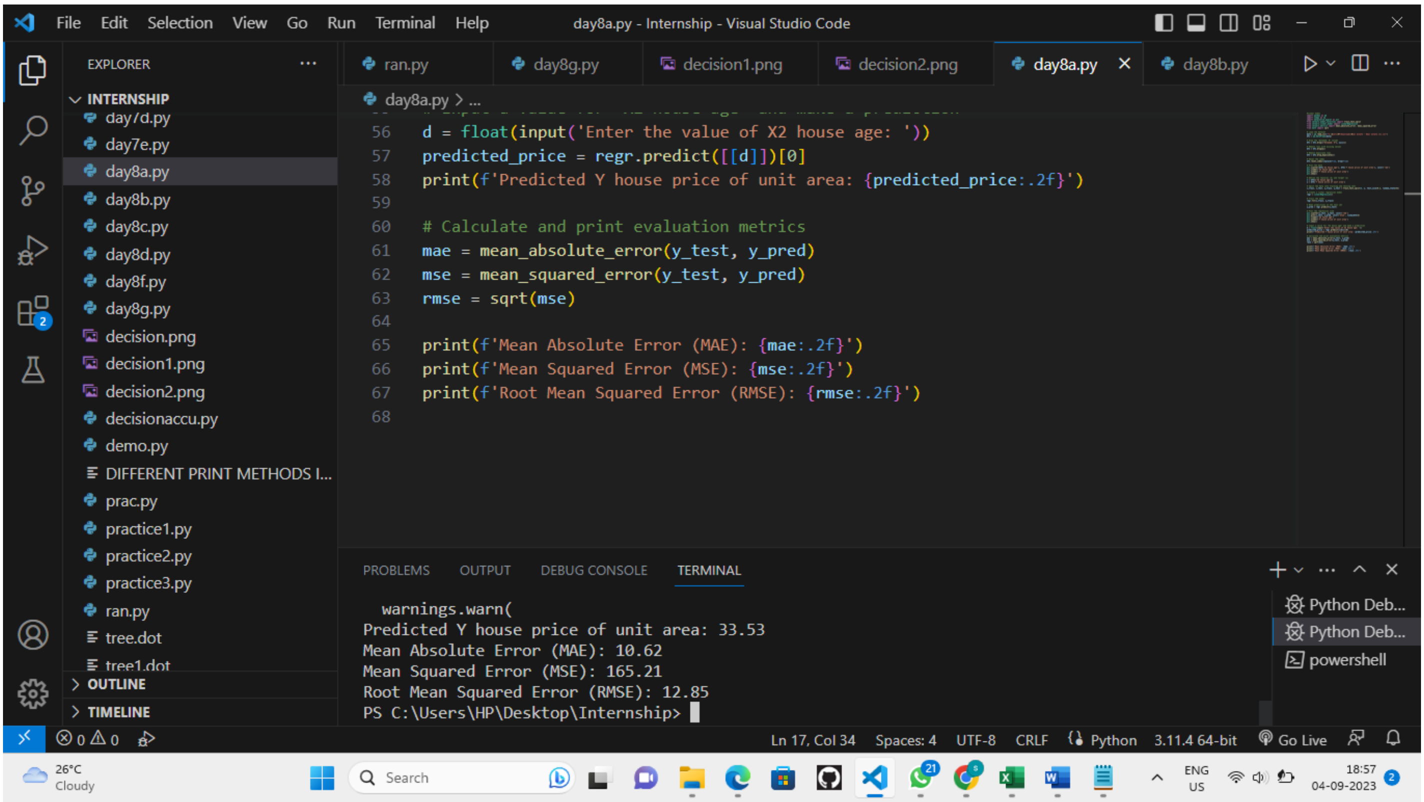
[redacted]



File Edit Selection View Go Run Terminal Help
day8a.py - Internship - Visual Studio Code
Figure 1
decision2.png
day8a.py
day8b.py
Y house price of unit area
X2 house age
lt
import train_test_split
ort LinearRegression
ean_absolute_error, mean_squared_error
\HP\Downloads\Real estate - Real estate (1).csv")
, axis=1)
lues
MINAL
ran.py
tree.dot
tree1.dot
OUTLINE
TIMELINE
PS C:\Users\HP\Desktop\Internship> c:; cd 'c:\Users\HP\Desktop\Internship'; & 'C:\Program Files\Python311\python.exe' 'c:\Users\HP\.vscode\extensions\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '59167' '--' 'C:\Users\HP\Desktop\Internship\day8a.py'
Python Deb...
Python Deb...
powershell
Ln 17, Col 34 Spaces: 4 UTF-8 CRLF Python 3.11.4 64-bit Go Live
26°C
Cloudy
Search
ENG
US
18:56
04-09-2023

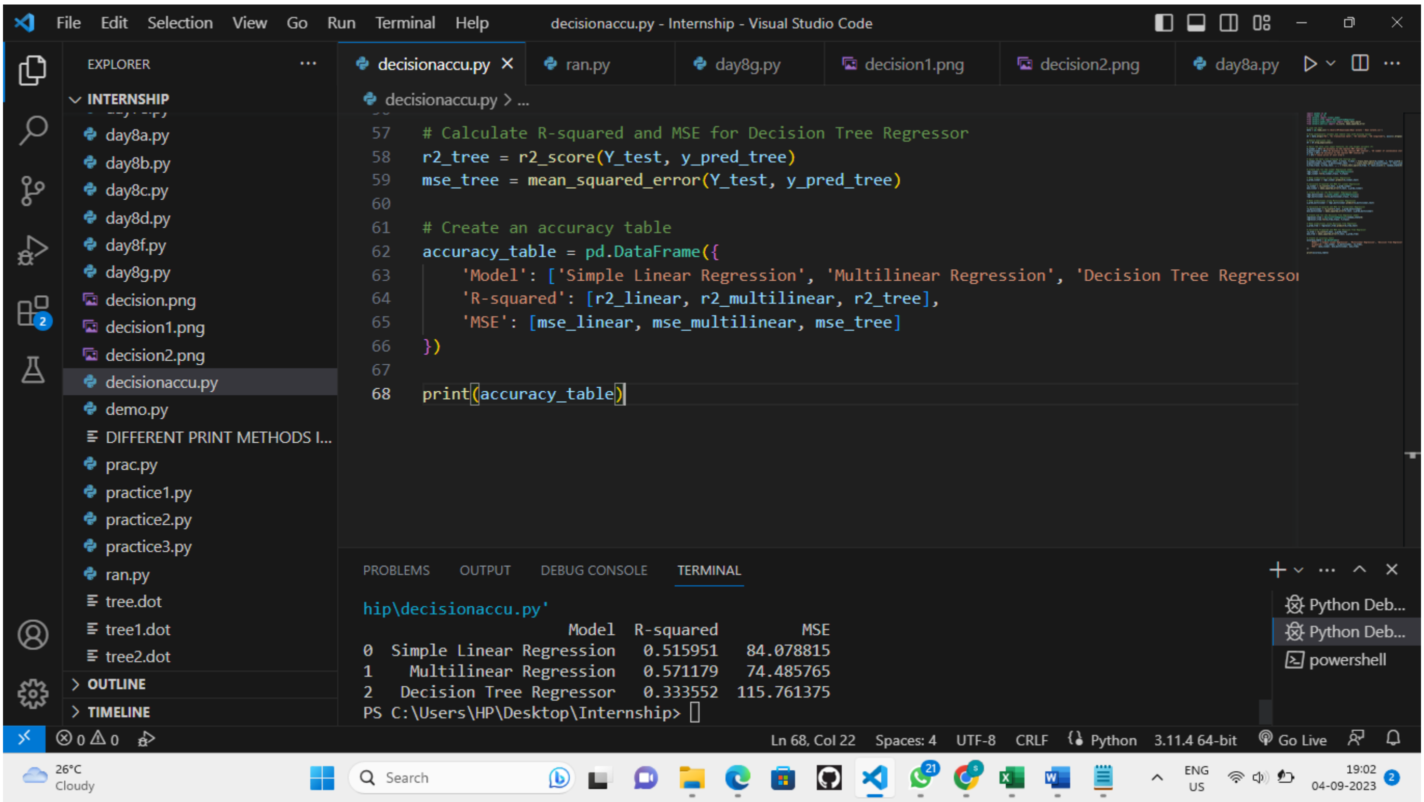


File Edit Selection View Go Run Terminal Help
day8a.py - Internship - Visual Studio Code
Figure 1
decision2.png
day8a.py
day8b.py
Y house price of unit area
X2 house age
x=27.49 y=34.2
lt
import train_test_split
ort LinearRegression
ean_absolute_error, mean_squared_error
\HP\Downloads\Real estate - Real estate (1).csv")
, axis=1)
lues
MINAL
ran.py
tree.dot
tree1.dot
OUTLINE
TIMELINE
PS C:\Users\HP\Desktop\Internship> c:; cd 'c:\Users\HP\Desktop\Internship'; & 'C:\Program Files\Python311\python.exe' 'c:\Users\HP\.vscode\extensions\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '59167' '--' 'C:\Users\HP\Desktop\Internship\day8a.py'
Python Deb...
Python Deb...
powershell
Ln 17, Col 34 Spaces: 4 UTF-8 CRLF Python 3.11.4 64-bit Go Live
26°C
Cloudy
Search
ENG
US
18:57
04-09-2023




```
import pandas as pd
import numpy as np
from sklearn import linear_model
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error

# 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_linear = df[['X3 distance to the nearest MRT station']]
X_multilinear = df[['X3 distance to the nearest MRT station', 'X4 number of convenience
stores']]
X_tree = 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_linear_train, X_linear_test, Y_train, Y_test = train_test_split(X_linear, Y,
test_size=0.2, random_state=0)
X_multilinear_train, X_multilinear_test, _, _ = train_test_split(X_multilinear, Y,
test_size=0.2, random_state=0)
X_tree_train, X_tree_test, _, _ = train_test_split(X_tree, Y, test_size=0.2, random_state=0)
```

```python
# Create and fit the Linear Regression model
regr_linear = linear_model.LinearRegression()
regr_linear.fit(X_linear_train, Y_train)

# Make predictions using Linear Regression
y_pred_linear = regr_linear.predict(X_linear_test)

# Calculate R-squared and MSE for Linear Regression
r2_linear = r2_score(Y_test, y_pred_linear)
mse_linear = mean_squared_error(Y_test, y_pred_linear)

# Create and fit the Multilinear Regression model
regr_multilinear = linear_model.LinearRegression()
regr_multilinear.fit(X_multilinear_train, Y_train)

# Make predictions using Multilinear Regression
y_pred_multilinear = regr_multilinear.predict(X_multilinear_test)

# Calculate R-squared and MSE for Multilinear Regression
r2_multilinear = r2_score(Y_test, y_pred_multilinear)
mse_multilinear = mean_squared_error(Y_test, y_pred_multilinear)

# Create and fit the Decision Tree Regressor model
regressor_tree = DecisionTreeRegressor(random_state=0)
regressor_tree.fit(X_tree_train, Y_train)

# Make predictions using Decision Tree Regressor
y_pred_tree = regressor_tree.predict(X_tree_test)

# Calculate R-squared and MSE for Decision Tree Regressor
r2_tree = r2_score(Y_test, y_pred_tree)
mse_tree = mean_squared_error(Y_test, y_pred_tree)

# Create an accuracy table
accuracy_table = pd.DataFrame({
    'Model': ['Simple Linear Regression', 'Multilinear Regression', 'Decision Tree
Regressor'],
    'R-squared': [r2_linear, r2_multilinear, r2_tree],
    'MSE': [mse_linear, mse_multilinear, mse_tree]
})

print(accuracy_table)
```



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.