**Predicting House Price Using Machine Learning Model**

Algorithm Code:

import pandas as pd

import numpy as np

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

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

from sklearn.preprocessing import OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

# Load training and test data

train\_data = pd.read\_csv('train.csv') # Training data file

test\_data = pd.read\_csv('test.csv') # Test data file

# Separate features (X) and target (y) for training data

X\_train = train\_data[['num\_rooms', 'sqft', 'neighborhood', 'other\_features', 'proximity\_to\_city']]

y\_train = train\_data['price']

# Separate features (X) and target (y) for test data

X\_test = test\_data[['num\_rooms', 'sqft', 'neighborhood', 'other\_features', 'proximity\_to\_city']]

y\_test = test\_data['price']

# Preprocessing pipeline for categorical features

preprocessor = ColumnTransformer(

transformers=[

('cat', OneHotEncoder(drop='first'), ['neighborhood', 'other\_features'])

],

remainder='passthrough' # Leave other features (like num\_rooms, sqft, proximity\_to\_city) as they are

)

# Function to evaluate models

def evaluate\_model(model, X\_train, X\_test, y\_train, y\_test):

# Fit the model

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

# Make predictions

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

# Calculate evaluation metrics

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

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

return mse, r2

# Dictionary to store results

results = {}

# Linear Regression

lr\_pipeline = Pipeline(steps=[

('preprocessor', preprocessor),

('model', LinearRegression())

])

mse, r2 = evaluate\_model(lr\_pipeline, X\_train, X\_test, y\_train, y\_test)

results['Linear Regression'] = {'MSE': mse, 'R2': r2}

# Decision Tree

dt\_pipeline = Pipeline(steps=[

('preprocessor', preprocessor),

('model', DecisionTreeRegressor(random\_state=42))

])

mse, r2 = evaluate\_model(dt\_pipeline, X\_train, X\_test, y\_train, y\_test)

results['Decision Tree'] = {'MSE': mse, 'R2': r2}

# Random Forest

rf\_pipeline = Pipeline(steps=[

('preprocessor', preprocessor),

('model', RandomForestRegressor(random\_state=42))

])

mse, r2 = evaluate\_model(rf\_pipeline, X\_train, X\_test, y\_train, y\_test)

results['Random Forest'] = {'MSE': mse, 'R2': r2}

# Display results

for model\_name, metrics in results.items():

print(f"{model\_name} - MSE: {metrics['MSE']:.2f}, R2: {metrics['R2']:.2f}")

# predictioin

new\_house\_data = pd.DataFrame({

'num\_rooms': [5], # Number of rooms

'sqft': [3375], # Square footage

'neighborhood': ['Rural'], # Neighborhood type (Urban or Rural)

'other\_features': ['No Garage'], # Garage feature (Has Garage or No Garage)

'proximity\_to\_city': [31] # Proximity to city (numeric value)

})

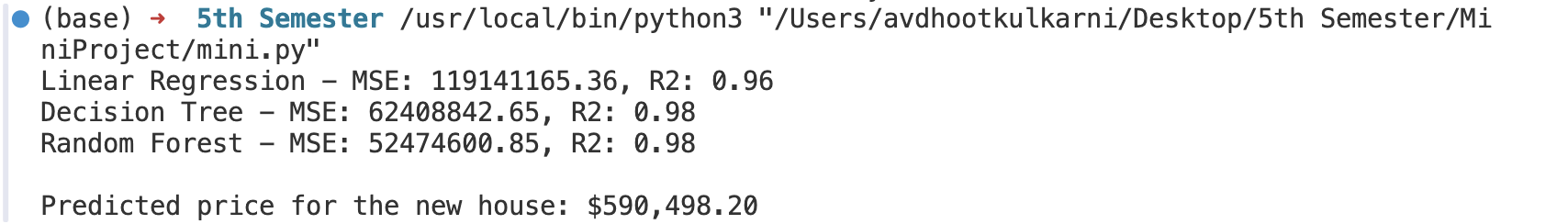
predicted\_price = rf\_pipeline.predict(new\_house\_data)

# Output the prediction

print("")

print(f"Predicted price for the new house: ${predicted\_price[0]:,.2f}")

Output:



This is the output based on the input given above in the code, this input can be changed.

# predictioin

new\_house\_data = pd.DataFrame({

'num\_rooms': [5], # Number of rooms

'sqft': [3375], # Square footage

'neighborhood': ['Rural'], # Neighborhood type (Urban or Rural)

'other\_features': ['No Garage'], # Garage feature (Has Garage or No Garage)

'proximity\_to\_city': [31] # Proximity to city (numeric value)

})

Based on different inputs required by the client, we can predict the House Price.