**Real Estate Price Prediction Using Machine Learning in Python**

Author: Agalya Sri S

Predicting real estate prices has become a crucial application of machine learning. With housing prices fluctuating due to various factors such as location, amenities, and market demand, a predictive model can help buyers, sellers, and investors make informed decisions. In this article, we will explore how to build a **real estate price prediction model** using **Python and Machine Learning (ML)**.

**Understanding the Dataset**

For this project, we will use a housing dataset containing features such as:

* **Transaction date**: Date of the property transaction.
* **House age**: Age of the property in years.
* **Distance to the nearest MRT station**: Proximity to the nearest Mass Rapid Transit station in meters.
* **Number of convenience stores**: Count of convenience stores in the vicinity.
* **Latitude and Longitude**: Geographical coordinates of the property.
* **House price of unit area**: The target variable representing the house price per unit area.

The dataset consists of **414 rows and 7 columns**.

This project utilizes machine learning techniques to estimate property prices per unit area based on multiple factors such as location, age, and proximity to amenities.

Download the dataset here: [Real\_Estate\_Dataset](https://statso.io/real-estate-prediction-case-study/#google_vignette)

**Importing Libraries**

import pandas as pd                 #data manipulation and analysis

import matplotlib.pyplot as plt     #visualization library

import seaborn as sns               #advance visualization library

#sklearn packages used to build and deploy the ML models

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

from sklearn.linear\_model import LinearRegression

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

**Loading Data**

Loading the dataset and understanding the datatypes by displaying the dataframe.

data = pd.read\_csv("/content/Real\_Estate\_Dataset.csv")

data

data.head()

data.info()

**Data Preprocessing**

Getting the descriptive statistics of the dataset which shows the mean, median, min – max values, etc.

data.describe()

Checking for missing values, in this scenario there are no missing values but for some other dataset we need to handle it separately.

data.isnull().sum()

Dropping the irrelevant columns, here the transaction date is not required for the prediction of the house price, so dropping it.

data = data.drop(['Transaction date'], axis=1)

**Data Visualization**

The histogram is used to plot the univariate analysis of each of the variables in the dataset. It is helpful in determining the frequency and the distribution of each of the variables.

# Creating subplots

fig, axes = plt.subplots(2, 3, figsize=(15, 10))

# Plot 1: Histogram of 'House Age'

sns.histplot(data['House age'], kde=True, ax=axes[0, 0])

axes[0, 0].set\_title('House Age Distribution')

# Plot 2: Histogram of 'Distance to the nearest MRT station'

sns.histplot(data['Distance to the nearest MRT station'], kde=True, ax=axes[0, 1])

axes[0, 1].set\_title('Distance to the nearest MRT station Distribution')

# Plot 3: Histogram of 'Number of convenience stores'

sns.histplot(data['Number of convenience stores'], kde=True, ax=axes[0, 2])

axes[0, 2].set\_title('Number of convenience stores Distribution')

# Plot 4: Histogram of 'Latitude'

sns.histplot(data['Latitude'], kde=True, ax=axes[1, 0])

axes[1, 0].set\_title('Latitude Distribution')

# Plot 5: Histogram of 'Longitude'

sns.histplot(data['Longitude'], kde=True, ax=axes[1, 1])

axes[1, 1].set\_title('Longitude Distribution')

# Plot 6: Histogram of 'House price of unit area'

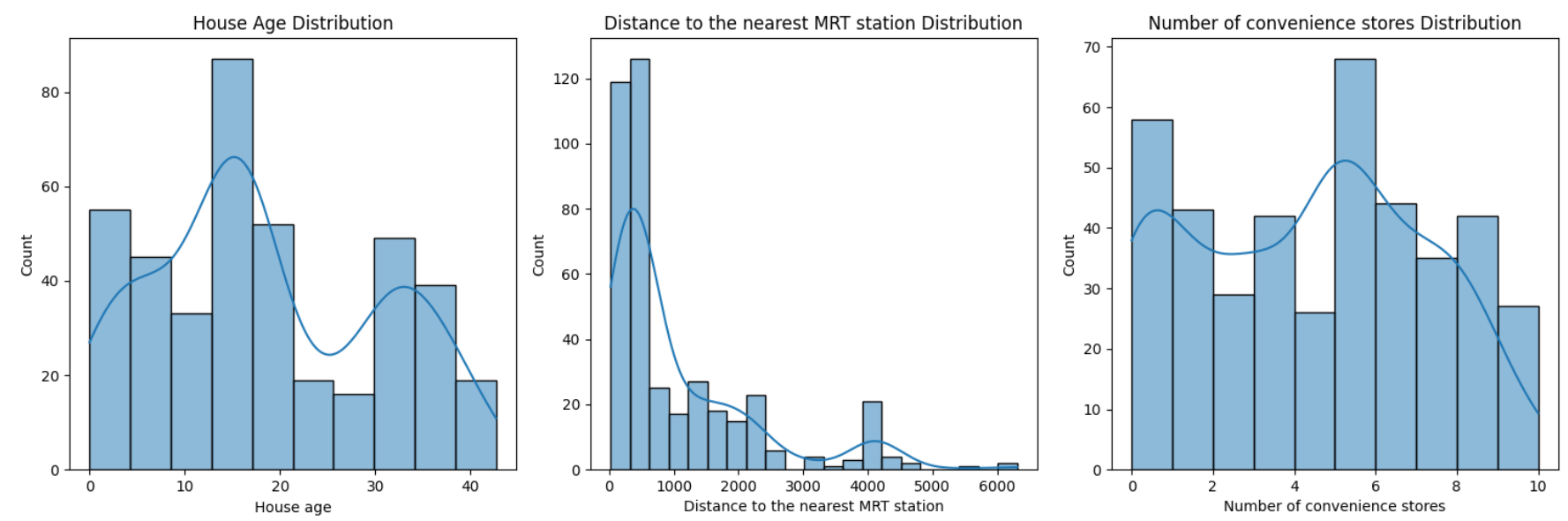
sns.histplot(data['House price of unit area'], kde=True, ax=axes[1, 2])

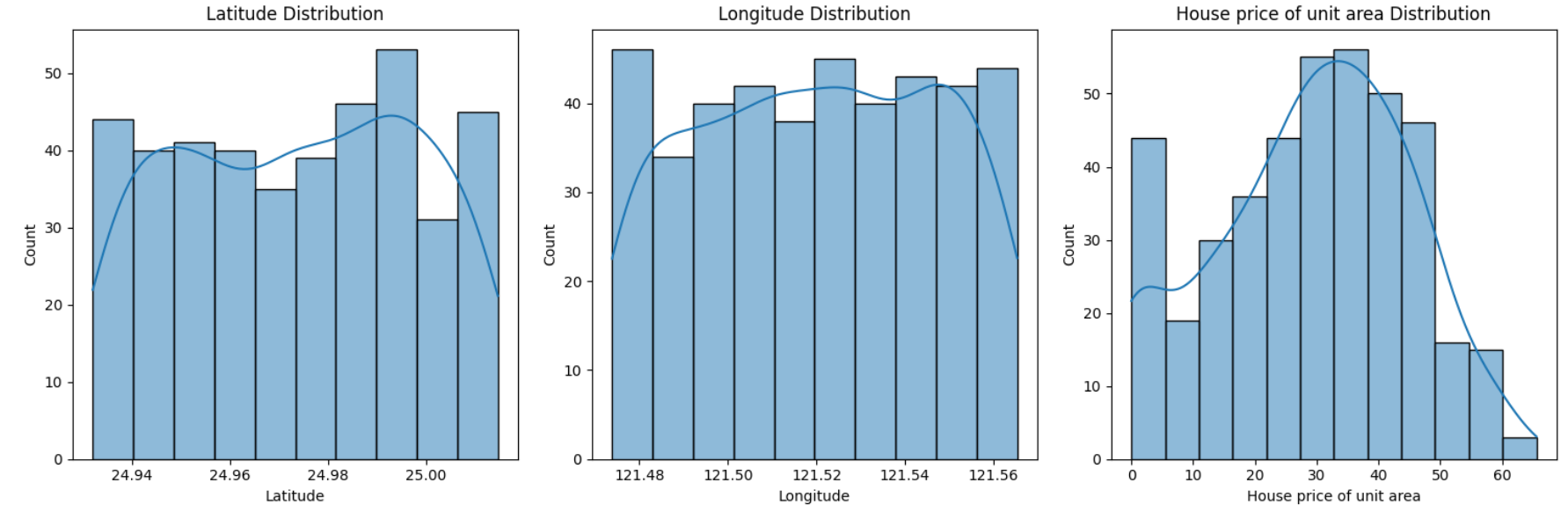
axes[1, 2].set\_title('House price of unit area Distribution')

# Adjust layout to prevent overlapping titles and labels

plt.tight\_layout()

plt.show()





The scatter plot is used to determine the bivariate analysis of the variables, it helps to determine the relationship between them.

fig, axes = plt.subplots(2, 3, figsize=(12, 8))

# Scatter plot 1

sns.scatterplot(x='House age', y='House price of unit area', data=data, ax=axes[1,0])

axes[1,0].set\_title('House Age vs. Price')

# Scatter plot 2

sns.scatterplot(x='Distance to the nearest MRT station', y='House price of unit area', data=data, ax=axes[1,1])

axes[1,1].set\_title('Distance to MRT vs. Price')

# Scatter plot 3

sns.scatterplot(x='Number of convenience stores', y='House price of unit area', data=data, ax=axes[1,2])

axes[1,2].set\_title('Convenience Stores vs. Price')

# Scatter plot 4

sns.scatterplot(x='Latitude', y='House price of unit area', data=data, ax=axes[0,0])

axes[0,0].set\_title('Latitude vs. Price')

# Scatter plot 5

sns.scatterplot(x='Longitude', y='House price of unit area', data=data, ax=axes[0,1])

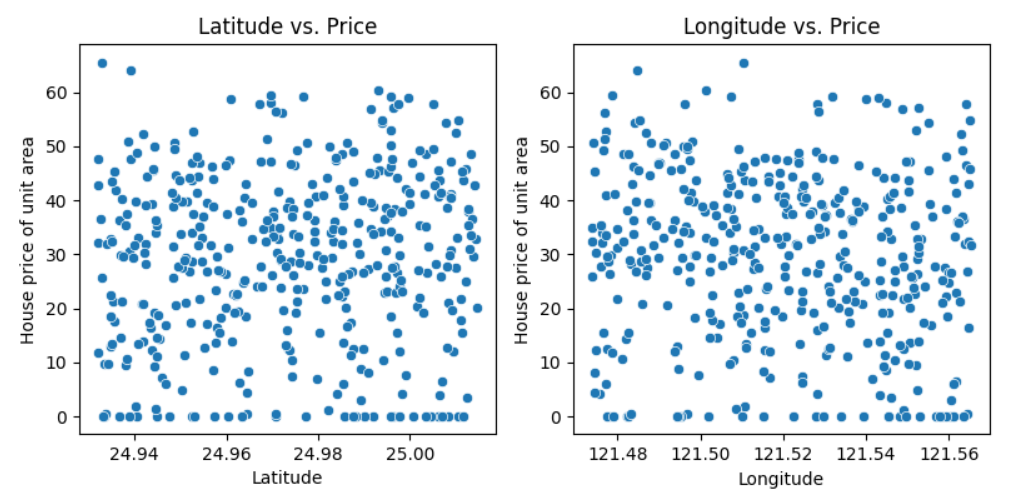
axes[0,1].set\_title('Longitude vs. Price')

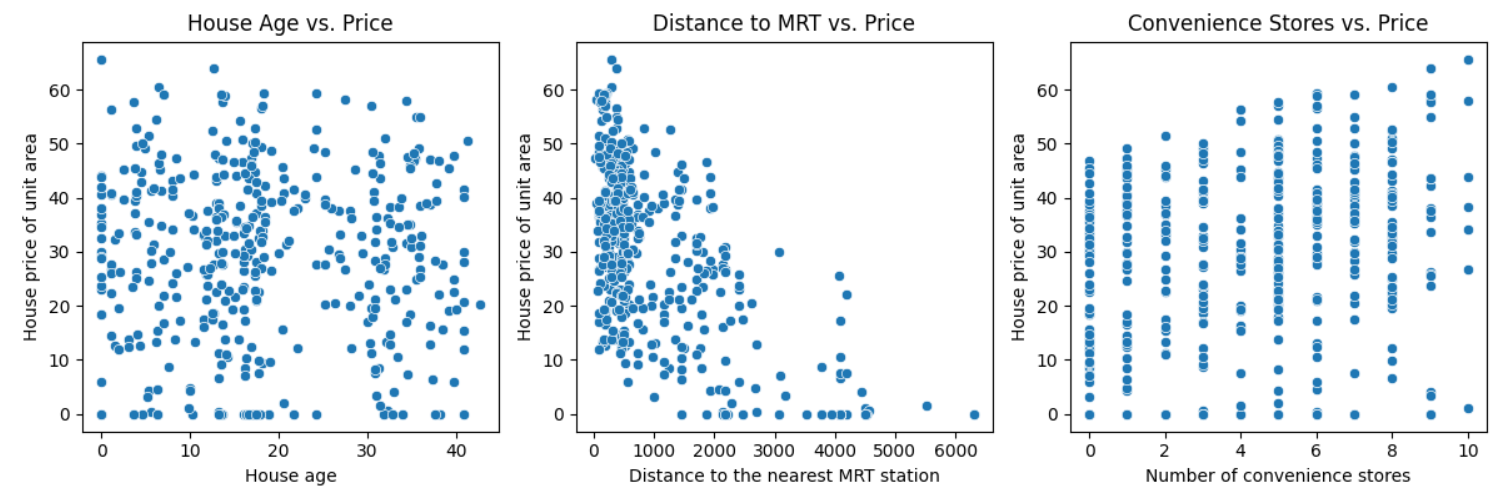
#removing the empty subplot

fig.delaxes(axes[0,2])

plt.tight\_layout()

plt.show()





The Correlation matrix is a multivariate analysis method used to determine how strong the relationship between the variables.

#the correlation matrix

correlation\_matrix = data.corr()

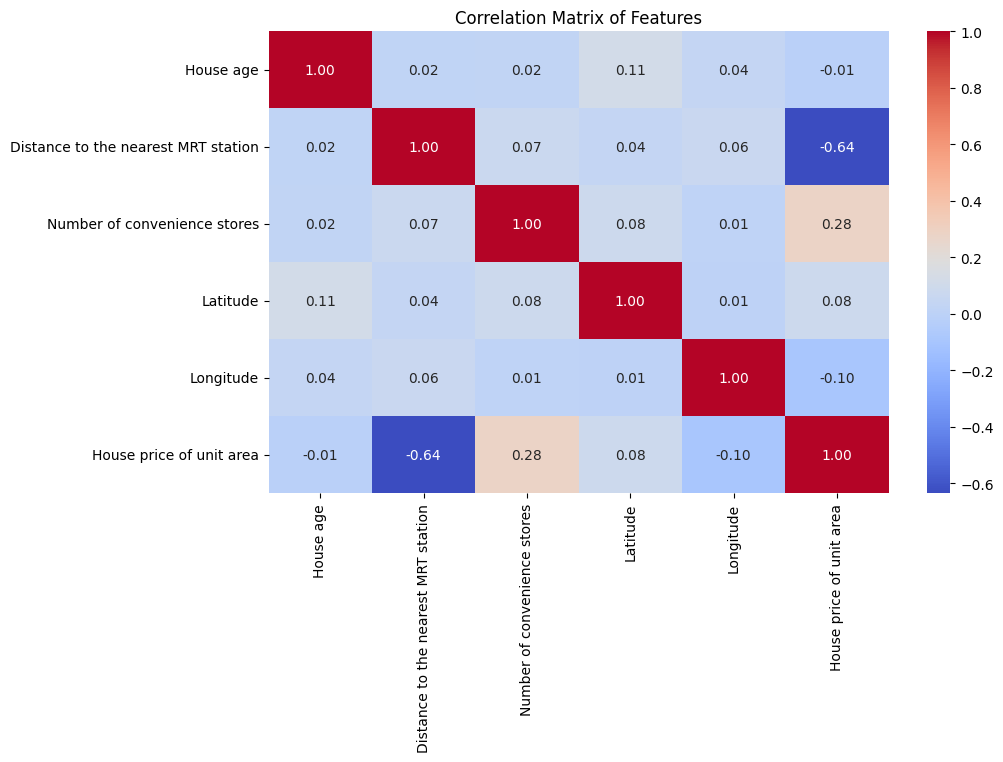
# Visualize the correlation matrix using a heatmap

plt.figure(figsize=(10, 6))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Correlation Matrix of Features')

plt.show()



**Model Building**

Based on the correlation coefficient taking the highly correlated variables as features and predicting the target. Splitting the dataset into training and test data in the ratio of 80:20. And using the Linear Regression model train the model using the training dataset.

# Selecting features and target variable

features = ['Distance to the nearest MRT station', 'Number of convenience stores', 'Latitude', 'Longitude']

target = 'House price of unit area'

X = data[features]

y = data[target]

# Splitting the dataset 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)

# Model initialization

model = LinearRegression()

# Training the model

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

Make predictions using the test data.

# Making predictions using the linear regression model

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

**Evaluation**

Evaluating the model using R-squared, MAE.

# Evaluate the model

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

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

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

print(f"R-squared: {r2}")

Visualizing actual vs predicted prices

# Visualize actual vs. predicted values

plt.figure(figsize=(6, 6))

plt.scatter(y\_test, y\_pred\_lr, alpha=0.5)  # Use alpha for better visualization if many points overlap

plt.plot([min(y\_test), max(y\_test)], [min(y\_test), max(y\_test)], color='red', linestyle='--') # Add a diagonal line for perfect prediction

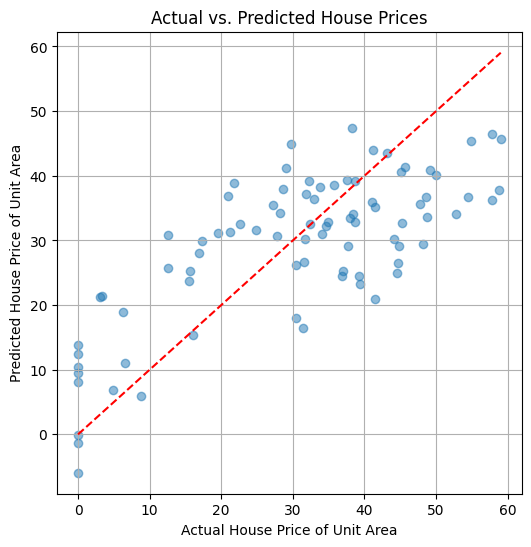
plt.xlabel("Actual House Price of Unit Area")

plt.ylabel("Predicted House Price of Unit Area")

plt.title("Actual vs. Predicted House Prices")

plt.grid(True)

plt.show()



**Conclusion**

* **Linear Regression** provides a basic yet effective approach to real estate price prediction.
* Further improvements can be made by trying more complex models like **XGBoost or Random Forest**.
* The dataset's features such as location, amenities, and accessibility significantly influence the pricing.

By implementing machine learning in real estate price prediction, we can build robust tools to help buyers, sellers, and investors make data-driven decisions.

**Next Steps:**

* For future work, testing models like Random Forest or XGBoost could enhance accuracy. Adding features such as property size and neighborhood ratings may further improve predictions.

💡 **Stay tuned for more insights on leveraging data science in real estate!**

**Would you like to explore more advanced ML models for price prediction? Let me know in the comments! 🚀**