Real Estate Price prediction

It plays a pivotal role in the real estate market and the broader economy. Accurate price predictions are essential for investors, developers, and homeowners, as they enable informed decision-making, strategic investment planning, and effective risk management.

The dataset provided comprises 414 entries, each containing detailed information about real estate transactions. It encompasses a range of features known to influence real estate pricing:

- Transaction date: Indicates the date of the property transaction.
- House age: Reflects the age of the property in years.
- Distance to the nearest MRT station: This factor is crucial for convenience and accessibility, measured in meters.
- Number of convenience stores: Represents the count of nearby convenience stores, which is indicative of the property's access to basic amenities.
- Latitude and Longitude: Geographical coordinates of the property, providing insight into its location.
- House price of unit area: The target variable, representing the house price per unit area.

The dataset is comprehensive, containing a combination of continuous and categorical variables. Importantly, it is devoid of missing values, rendering it robust for predictive modeling purposes.

The primary aim is to develop a predictive model capable of accurately forecasting the house price per unit area based on various features such as property age, proximity to key amenities (MRT stations and convenience stores), and geographical location.

In [62]:
```
import pandas as pd
```

The dataset comprises 7 columns, each providing specific information about real estate transactions. Here's a concise summary of each column:

1. **Transaction date**: This column records the date of the real estate transaction.
2. **House age**: Indicates the age of the house in years at the time of the transaction.
3. **Distance to the nearest MRT station**: Specifies the distance, in meters, from the property to the nearest Mass Rapid Transit (MRT) station. This metric is crucial for assessing convenience and accessibility.
4. **Number of convenience stores**: Represents the count of convenience stores located in the vicinity of the property. This factor is significant as it reflects the property's proximity to basic amenities.
5. **Latitude**: Provides the geographical coordinate of the property's location in terms of latitude. Latitude values denote the north-south position on the Earth's surface.
6. **Longitude**: Indicates the geographical coordinate of the property's location in terms of longitude. Longitude values denote the east-west position on the Earth's surface.
7. **House price of unit area**: This column specifies the price of the house per unit area, which is a key target variable for predictive modeling. It represents the price at which the property was sold per unit of its total area.

These columns collectively provide comprehensive information about each real estate transaction, facilitating analysis and modeling to predict house prices accurately.

In [63]:
```
# Display the first few rows of the dataset and the info about the dataset
real_estate_data_head = real_estate_data.head()
```

```
data_info = real_estate_data.info()

print(real_estate_data_head)
print(data_info)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 414 entries, 0 to 413
Data columns (total 7 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Transaction date                        414 non-null    object
 1   House age                               414 non-null    float64
 2   Distance to the nearest MRT station     414 non-null    float64
 3   Number of convenience stores            414 non-null    int64
 4   Latitude                                414 non-null    float64
 5   Longitude                               414 non-null    float64
 6   House price of unit area                414 non-null    float64
dtypes: float64(5), int64(1), object(1)
memory usage: 22.8+ KB
             Transaction date  House age  Distance to the nearest MRT
station  \
0  2012-09-02 16:42:30.519336       13.3
4082.0150
1  2012-09-04 22:52:29.919544       35.5
274.0144
2  2012-09-05 01:10:52.349449        1.1
1978.6710
3  2012-09-05 13:26:01.189083       22.2
1055.0670
4  2012-09-06 08:29:47.910523        8.5
967.4000

   Number of convenience stores   Latitude   Longitude  \
0                             8  25.007059  121.561694
1                             2  25.012148  121.546990
2                            10  25.003850  121.528336
3                             5  24.962887  121.482178
4                             6  25.011037  121.479946

   House price of unit area
0                  6.488673
1                 24.970725
2                 26.694267
3                 38.091638
4                 21.654710
None
```

In [64]:

```
print(real_estate_data.isnull().sum())
Transaction date                        0
House age                               0
```

```
Distance to the nearest MRT station    0
Number of convenience stores           0
Latitude                               0
Longitude                              0
House price of unit area               0
dtype: int64
```

In [65]:

```
# Descriptive statistics of the dataset
descriptive_stats = real_estate_data.describe()

print(descriptive_stats)
```

```
        House age  Distance to the nearest MRT station  \
count  414.000000                          414.000000
mean    18.405072                         1064.468233
std     11.757670                         1196.749385
min      0.000000                           23.382840
25%      9.900000                          289.324800
50%     16.450000                          506.114400
75%     30.375000                         1454.279000
max     42.700000                         6306.153000

       Number of convenience stores    Latitude   Longitude  \
count                    414.000000  414.000000  414.000000
mean                       4.265700   24.973605  121.520268
std                        2.880498    0.024178    0.026989
min                        0.000000   24.932075  121.473888
25%                        2.000000   24.952422  121.496866
50%                        5.000000   24.974353  121.520912
75%                        6.750000   24.994947  121.544676
max                       10.000000   25.014578  121.565321

       House price of unit area
count                414.000000
mean                  29.102149
std                   15.750935
min                    0.000000
25%                   18.422493
50%                   30.394070
75%                   40.615184
max                   65.571716
```

In [66]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [67]:

```
# Set the aesthetic style of the plots
sns.set_style("whitegrid")
```

The histograms provide valuable insights into the distribution of each variable:

1. **House Age**: The distribution appears relatively uniform, with a slight increase in the number of newer properties (lower age), suggesting a diverse range of property ages within the dataset.
2. **Distance to the Nearest MRT Station**: Most properties are situated in close proximity to an MRT station, evidenced by the high frequency of lower distances. However, there is a long tail extending towards higher distances, indicating the presence of properties located farther away from MRT stations.
3. **Number of Convenience Stores**: This histogram displays a wide range of counts, with notable peaks at specific counts such as 0, 5, and 10 convenience stores. This pattern suggests common configurations regarding the availability of convenience stores near the properties.
4. **Latitude and Longitude**: Both distributions show concentrations, indicating that the properties are situated within a limited geographic area. This suggests that the dataset encompasses properties located in a specific region.
5. **House Price of Unit Area**: The histogram reveals a right-skewed distribution, with a concentration of properties in the lower price range and fewer properties as prices increase. This distribution pattern highlights the variability in property prices within the dataset, with a notable number of properties priced at the lower end.

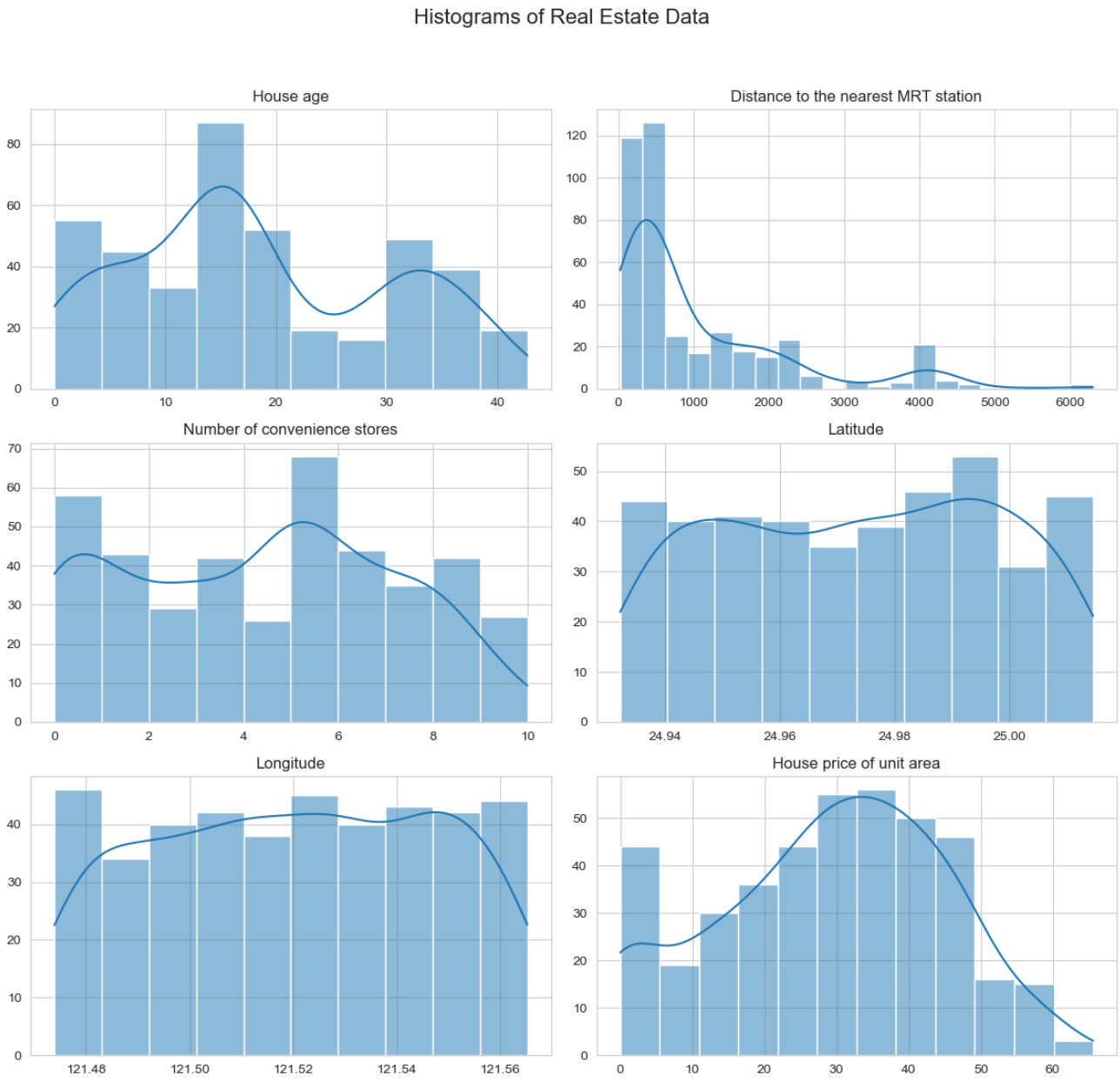
In [68]:

```
# Create histograms for the numerical columns
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(12, 12))
fig.suptitle('Histograms of Real Estate Data', fontsize=16)

cols = ['House age', 'Distance to the nearest MRT station', 'Number of
convenience stores',
        'Latitude', 'Longitude', 'House price of unit area']

for i, col in enumerate(cols):
    sns.histplot(real_estate_data[col], kde=True, ax=axes[i//2, i%2])
    axes[i//2, i%2].set_title(col)
    axes[i//2, i%2].set_xlabel('')
    axes[i//2, i%2].set_ylabel('')

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



The scatter plots revealed intriguing relationships between various factors and house prices:

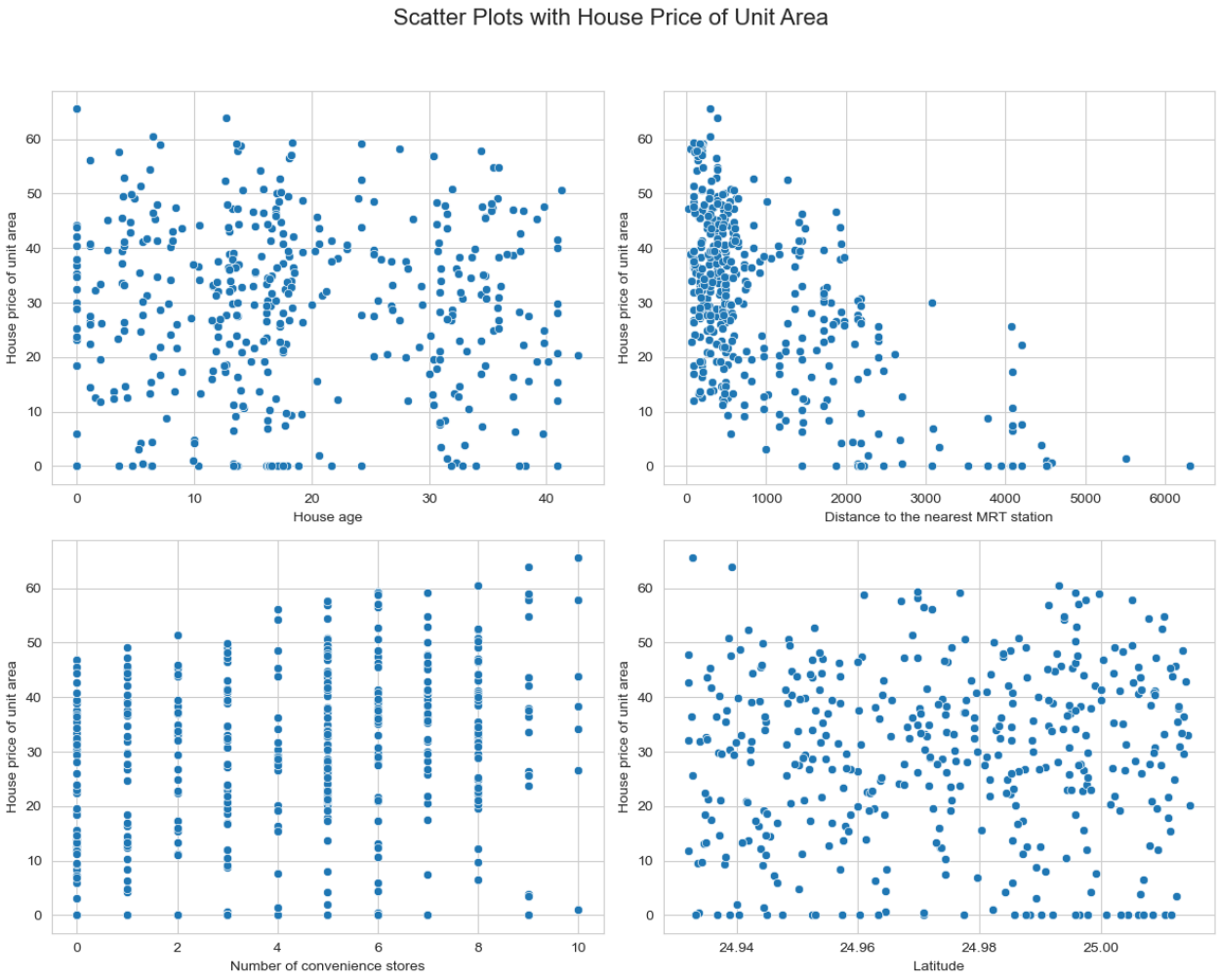
1. **House Age vs. House Price**: No strong linear relationship is evident between house age and price. However, there appears to be a trend where very new and very old houses might command higher prices.
2. **Distance to the Nearest MRT Station vs. House Price**: A clear trend emerges, indicating that as the distance to the nearest MRT station increases, house prices tend to decrease. This observation suggests a strong negative relationship between these two variables.
3. **Number of Convenience Stores vs. House Price**: A positive relationship seems to exist between the number of convenience stores and house prices. Properties located in areas with more convenience stores nearby tend to have higher prices.
4. **Latitude vs. House Price**: While not demonstrating a strong linear relationship, there appears to be a discernible pattern where certain latitudes correspond to higher or lower house prices. This pattern might reflect the desirability of specific neighborhoods or locations.

In [69]:

```
# Scatter plots to observe the relationship with house price
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 10))
fig.suptitle('Scatter Plots with House Price of Unit Area', fontsize=16)

# Scatter plot for each variable against the house price
sns.scatterplot(data=real_estate_data, x='House age', y='House price of
unit area', ax=axes[0, 0])
sns.scatterplot(data=real_estate_data, x='Distance to the nearest MRT
station', y='House price of unit area', ax=axes[0, 1])
sns.scatterplot(data=real_estate_data, x='Number of convenience stores',
y='House price of unit area', ax=axes[1, 0])
sns.scatterplot(data=real_estate_data, x='Latitude', y='House price of unit
area', ax=axes[1, 1])

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



The correlation matrix offers quantified insights into the relationships between each variable, particularly concerning house prices:

1. **House Age**: Exhibits a very weak negative correlation with house price (-0.012), suggesting that age is not a strong predictor of price in this dataset.

2. **Distance to Nearest MRT Station**: Demonstrates a strong negative correlation with house price (-0.637). This implies that properties closer to MRT stations tend to command higher prices, underscoring the significance of proximity to public transportation in property valuation.
3. **Number of Convenience Stores**: Shows a moderate positive correlation with house price (0.281). The presence of more convenience stores in the vicinity appears to positively influence property prices.
4. **Latitude and Longitude**: Both variables display weak correlations with house prices. Latitude exhibits a slight positive correlation (0.081), whereas longitude has a slight negative correlation (-0.099).

Overall, the most influential factors impacting house prices in this dataset are the proximity to MRT stations and the number of convenience stores nearby. Geographic location (latitude and longitude) and the age of the house appear to have relatively less impact on property prices.

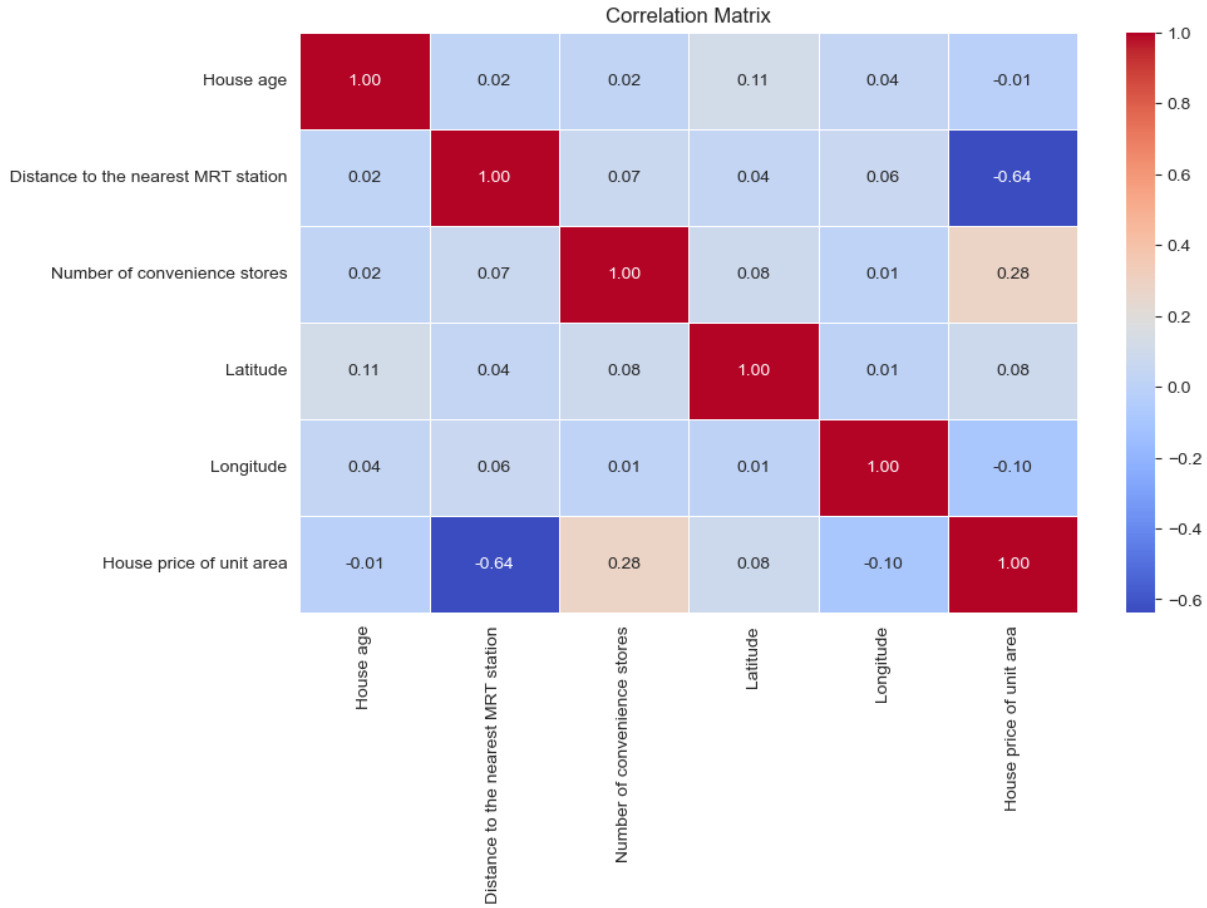
In [70]:

```
# Correlation matrix
correlation_matrix = real_estate_data.corr()

# Plotting the correlation matrix
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",
linewidths=.5)
plt.title('Correlation Matrix')
plt.show()

print(correlation_matrix)
C:\Users\anike\AppData\Local\Temp\ipykernel_13536\853538187.py:2:
FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a
future version, it will default to False. Select only valid columns or
specify the value of numeric_only to silence this warning.
```



	House age
House age	1.000000
Distance to the nearest MRT station	0.021596
Number of convenience stores	0.021973
Latitude	0.114345
Longitude	0.036449
House price of unit area	-0.012284

	Distance to the nearest MRT station
House age	0.021596
Distance to the nearest MRT station	1.000000
Number of convenience stores	0.069015
Latitude	0.038954
Longitude	0.064229
House price of unit area	-0.636579

	Number of convenience stores	Latitude
House age	0.021973	0.114345
Distance to the nearest MRT station	0.069015	0.038954
Number of convenience stores	1.000000	0.082725
Latitude	0.082725	1.000000
Longitude	0.013156	0.007754
House price of unit area	0.280763	0.081008

```
                                    Longitude  House price of unit area
House age                            0.036449                 -0.012284
Distance to the nearest MRT station  0.064229                 -0.636579
Number of convenience stores         0.013156                  0.280763
Latitude                             0.007754                  0.081008
Longitude                            1.000000                 -0.098626
House price of unit area            -0.098626                  1.000000
```

In [71]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

In [72]:

```
# Selecting features and target variable
features = ['Distance to the nearest MRT station', 'Number of convenience
stores', 'Latitude', 'Longitude']
target = 'House price of unit area'
```

In [73]:

```
X = real_estate_data[features]
y = real_estate_data[target]
```

In [74]:

```
# 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)
```

In [75]:

```
# Model initialization
model = LinearRegression()
```

In [76]:

```
# Training the model
model.fit(X_train, y_train)
```

Out[76]:

```
LinearRegression
LinearRegression()
```

A clear interpretation of the scatter plot representing the model's predictions. It effectively communicates the following observations:

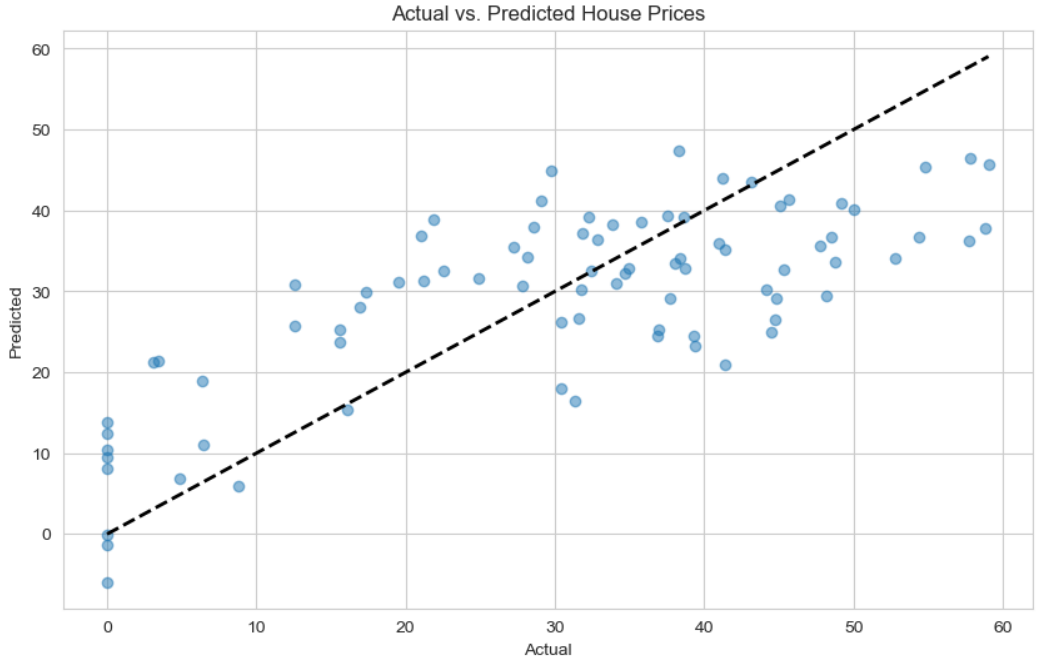
1. **Proximity to Diagonal Line**: Many points are close to the diagonal line, indicating that the model makes reasonably accurate predictions for a substantial portion of the test set. This suggests that the model performs well in predicting house prices accurately in certain instances.
2. **Deviation from Diagonal Line**: Some points are further from the diagonal line, highlighting areas where the model's predictions deviate more significantly from the actual values. These deviations may signify instances where the model struggles to accurately predict house prices, possibly due to complexities or outliers in the data.

The conclusion succinctly summarizes the process of predicting real estate prices using machine learning with Python. Overall, the explanation provides a concise overview of the predictive modeling process and its implications for real estate price prediction.

In [77]:

```
# Making predictions using the linear regression model
y_pred_lr = model.predict(X_test)
```

```
# Visualization: Actual vs. Predicted values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred_lr, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--',
lw=2)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs. Predicted House Prices')
plt.show()
```



Dash is an open-source Python framework used for building interactive web applications. It's particularly popular for creating analytical and data visualization applications due to its integration with popular data science libraries like Plotly and Pandas.

In this code snippet:

Dash is the main Dash library. html and dcc (Dash Core Components) are used to create HTML and interactive components. Input, Output, and State are used for creating callbacks in Dash, enabling interactivity.

In [78]:

```
import dash
from dash import html, dcc, Input, Output, State
import pandas as pd
```

Dash simplifies the process of creating web-based data applications in Python, enabling developers and data scientists to focus on building rich and interactive user experiences without having to delve deeply into web development technologies.

In [79]:

```
# Initialize the Dash app
app = dash.Dash(__name__)
```

Here, we define the HTML structure of the app using Dash's HTML components. The layout includes a title (html.H1), input fields for distance to the MRT station, number of convenience stores, latitude, and longitude (dcc.Input), and a button to trigger the prediction (html.Button).

In [80]:

```
# Define the layout of the app
app.layout = html.Div([
    html.Div([
        html.H1("Real Estate Price Prediction", style={'text-align':
'center'}),

        html.Div([
            dcc.Input(id='distance_to_mrt', type='number',
placeholder='Distance to MRT Station (meters)',
                      style={'margin': '10px', 'padding': '10px'}),
            dcc.Input(id='num_convenience_stores', type='number',
placeholder='Number of Convenience Stores',
                      style={'margin': '10px', 'padding': '10px'}),
            dcc.Input(id='latitude', type='number', placeholder='Latitude',
                      style={'margin': '10px', 'padding': '10px'}),
            dcc.Input(id='longitude', type='number',
placeholder='Longitude',
                      style={'margin': '10px', 'padding': '10px'}),
            html.Button('Predict Price', id='predict_button', n_clicks=0,
                        style={'margin': '10px', 'padding': '10px',
'background-color': '#007BFF', 'color': 'white'}),
        ], style={'text-align': 'center'}),

        html.Div(id='prediction_output', style={'text-align': 'center',
'font-size': '20px', 'margin-top': '20px'})
    ], style={'width': '50%', 'margin': '0 auto', 'border': '2px solid
#007BFF', 'padding': '20px', 'border-radius': '10px'})
])
```

This is a callback function that updates the output (prediction result) when the 'Predict Price' button is clicked. Output('prediction_output', 'children') indicates that the inner content (children) of the component with id prediction_output will be updated by this callback.

The callback takes the number of button clicks as Input and the values of the four input fields as State. The function update_output is executed when the button is clicked, using the input values to generate a prediction.

Inside the update_output function, the inputs are first checked to ensure they are not None. The inputs are then arranged into a Pandas DataFrame, matching the expected format for the model. The model.predict method is called to generate a prediction. This assumes that a trained model named model exists and is accessible within this script. The function returns either the predicted price or a prompt to enter all values.

In [81]:

```
# Define callback to update output
@app.callback(
    Output('prediction_output', 'children'),
    [Input('predict_button', 'n_clicks')],
    [State('distance_to_mrt', 'value'),
     State('num_convenience_stores', 'value'),
     State('latitude', 'value'),
     State('longitude', 'value')]
)
```

```
def update_output(n_clicks, distance_to_mrt, num_convenience_stores,
latitude, longitude):
    if n_clicks > 0 and all(v is not None for v in [distance_to_mrt,
num_convenience_stores, latitude, longitude]):
        # Prepare the feature vector
        features = pd.DataFrame([[distance_to_mrt, num_convenience_stores,
latitude, longitude]],
                                columns=['distance_to_mrt',
'num_convenience_stores', 'latitude', 'longitude'])
        # Predict
        prediction = model.predict(features)[0]
        return f'Predicted House Price of Unit Area: {prediction:.2f}'
    elif n_clicks > 0:
        return 'Please enter all values to get a prediction'
    return ''
```

This part runs the app server when the script is executed directly (**name** == '**main**'). debug=True enables the debug mode, which provides an interactive debugger in the browser and auto-reloads the server on code changes.

In [82]:

```
# Run the app
if __name__ == '__main__':
    app.run_server(debug=True)
```

Conclusion

Real Estate Price Prediction, within the realm of Machine Learning, entails the task of using algorithms and statistical techniques to estimate or forecast the future prices of real estate properties, encompassing various types such as houses, apartments, or commercial buildings. The primary objective is to furnish accurate property rates to buyers, sellers, investors, and real estate professionals, enabling them to make well-informed decisions regarding real estate transactions. Employing Python for this purpose allows for the utilization of robust libraries and frameworks for data processing, model building, and predictive analysis. By leveraging Machine Learning algorithms, predictive models can be trained on historical real estate data to discern patterns and relationships among different features, ultimately yielding predictions of future property prices. This prediction provids an overview of the Real Estate Price Prediction process with Machine Learning using Python, elucidating the significance of accurate predictions in the real estate market. Feel free to pose any insightful inquiries or share thoughts in the comments section below, fostering a collaborative discourse on this intriguing domain.