In [3]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
# Load the dataset
data = pd.read_excel('/Users/user/Downloads/Real_estate.xlsx')
```

In [5]:
```
# Display the first few rows of the data
print(data.head())
```

```
   No  X1 transaction date  X2 house age  \
0   1             2012.916667          32.0
1   2             2012.916667          19.5
2   3             2013.583333          13.3
3   4             2013.500000          13.3
4   5             2012.833333           5.0

   X3 distance to the nearest MRT station  X4 number of convenience stores  \
0                                84.87882                               10
1                               306.59470                                9
2                               561.98450                                5
3                               561.98450                                5
4                               390.56840                                5

   X5 latitude  X6 longitude  Y house price of unit area
0     24.98298     121.54024                        37.9
1     24.98034     121.53951                        42.2
2     24.98746     121.54391                        47.3
3     24.98746     121.54391                        54.8
4     24.97937     121.54245                        43.1
```

In [6]:
```
# Check for missing values
print(data.isnull().sum())
```

```
No                                      0
X1 transaction date                     0
X2 house age                            0
X3 distance to the nearest MRT station  0
X4 number of convenience stores         0
X5 latitude                             0
X6 longitude                            0
Y house price of unit area              0
dtype: int64
```

In [7]:
```
# Basic statistics of the dataset
print(data.describe())
```

	No	X1 transaction date	X2 house age
count	414.000000	414.000000	414.000000
mean	207.500000	2013.148953	17.712560
std	119.655756	0.281995	11.392485
min	1.000000	2012.666667	0.000000
25%	104.250000	2012.916667	9.025000
50%	207.500000	2013.166667	16.100000
75%	310.750000	2013.416667	28.150000
max	414.000000	2013.583333	43.800000

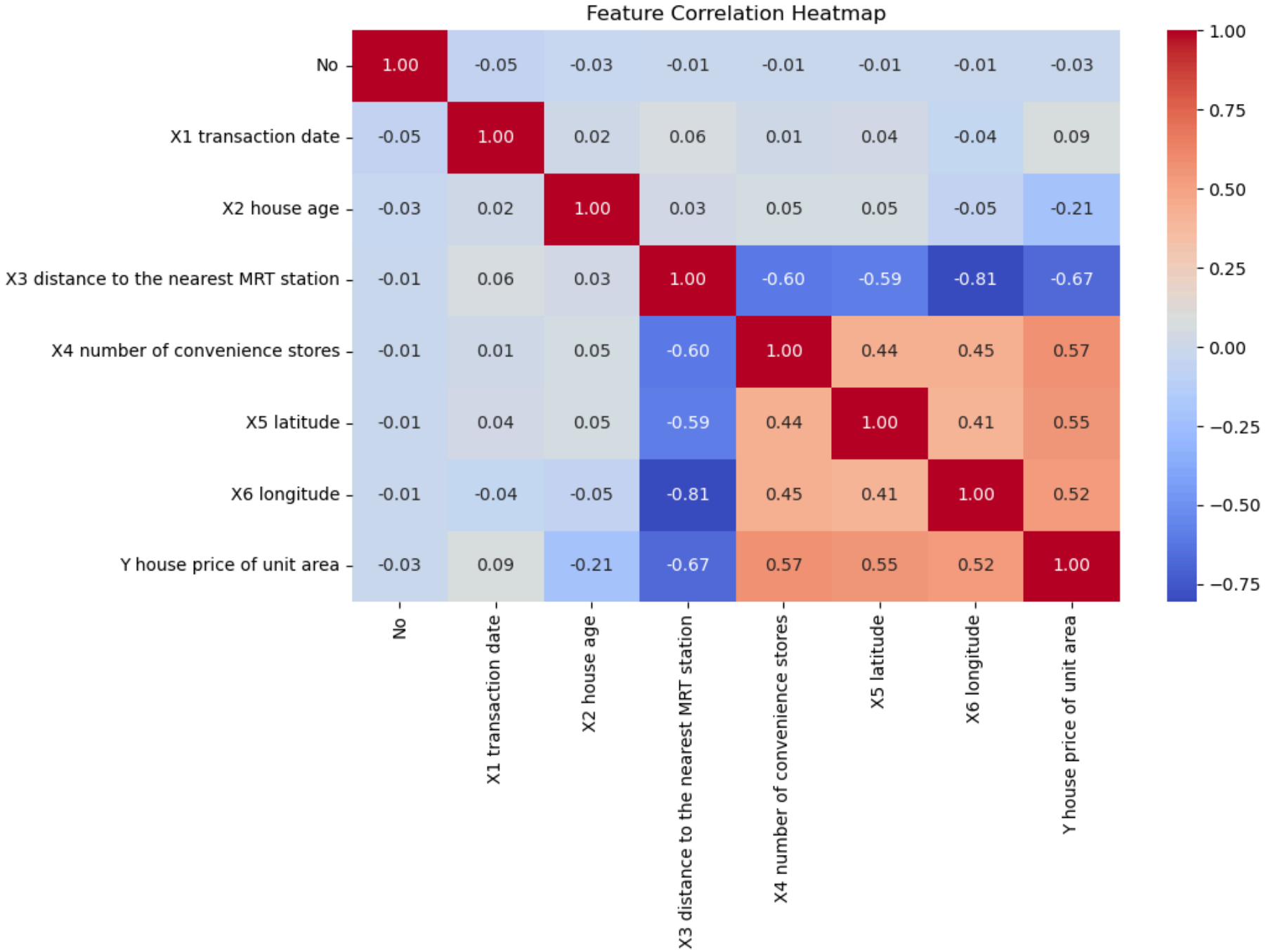
	X3 distance to the nearest MRT station
count	414.000000
mean	1083.885689
std	1262.109595
min	23.382840
25%	289.324800
50%	492.231300
75%	1454.279000
max	6488.021000

	X4 number of convenience stores	X5 latitude	X6 longitude
count	414.000000	414.000000	414.000000
mean	4.094203	24.969030	121.533361
std	2.945562	0.012410	0.015347
min	0.000000	24.932070	121.473530
25%	1.000000	24.963000	121.528085
50%	4.000000	24.971100	121.538630
75%	6.000000	24.977455	121.543305
max	10.000000	25.014590	121.566270

	Y house price of unit area
count	414.000000
mean	37.980193
std	13.606488
min	7.600000
25%	27.700000
50%	38.450000
75%	46.600000
max	117.500000

In [8]:
```
# Assuming the target variable (e.g., house price) is the last column
# Adjust `target_column` to the actual column name for house prices in the dataset.
target_column = 'Y house price of unit area'
features = data.drop(target_column, axis=1)
target = data[target_column]
```

In [9]:
```
# Optional: visualize correlations
plt.figure(figsize=(10, 6))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Feature Correlation Heatmap")
plt.show()
```


Feature Correlation Heatmap
No
X1 transaction date
X2 house age
X3 distance to the nearest MRT station
X4 number of convenience stores
X5 latitude
X6 longitude
Y house price of unit area
1.00 -0.05 -0.03 -0.01 -0.01 -0.01 -0.01 -0.03
-0.05 1.00 0.02 0.06 0.01 0.04 -0.04 0.09
-0.03 0.02 1.00 0.03 0.05 0.05 -0.05 -0.21
-0.01 0.06 0.03 1.00 -0.60 -0.59 -0.81 -0.67
-0.01 0.01 0.05 -0.60 1.00 0.44 0.45 0.57
-0.01 0.04 0.05 -0.59 0.44 1.00 0.41 0.55
-0.01 -0.04 -0.05 -0.81 0.45 0.41 1.00 0.52
-0.03 0.09 -0.21 -0.67 0.57 0.55 0.52 1.00
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

In [10]:
```
# Split the data into training and test sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)
```

In [11]:
```
# Initialize and train the model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

Out[11]:
```
RandomForestRegressor(random_state=42)
```

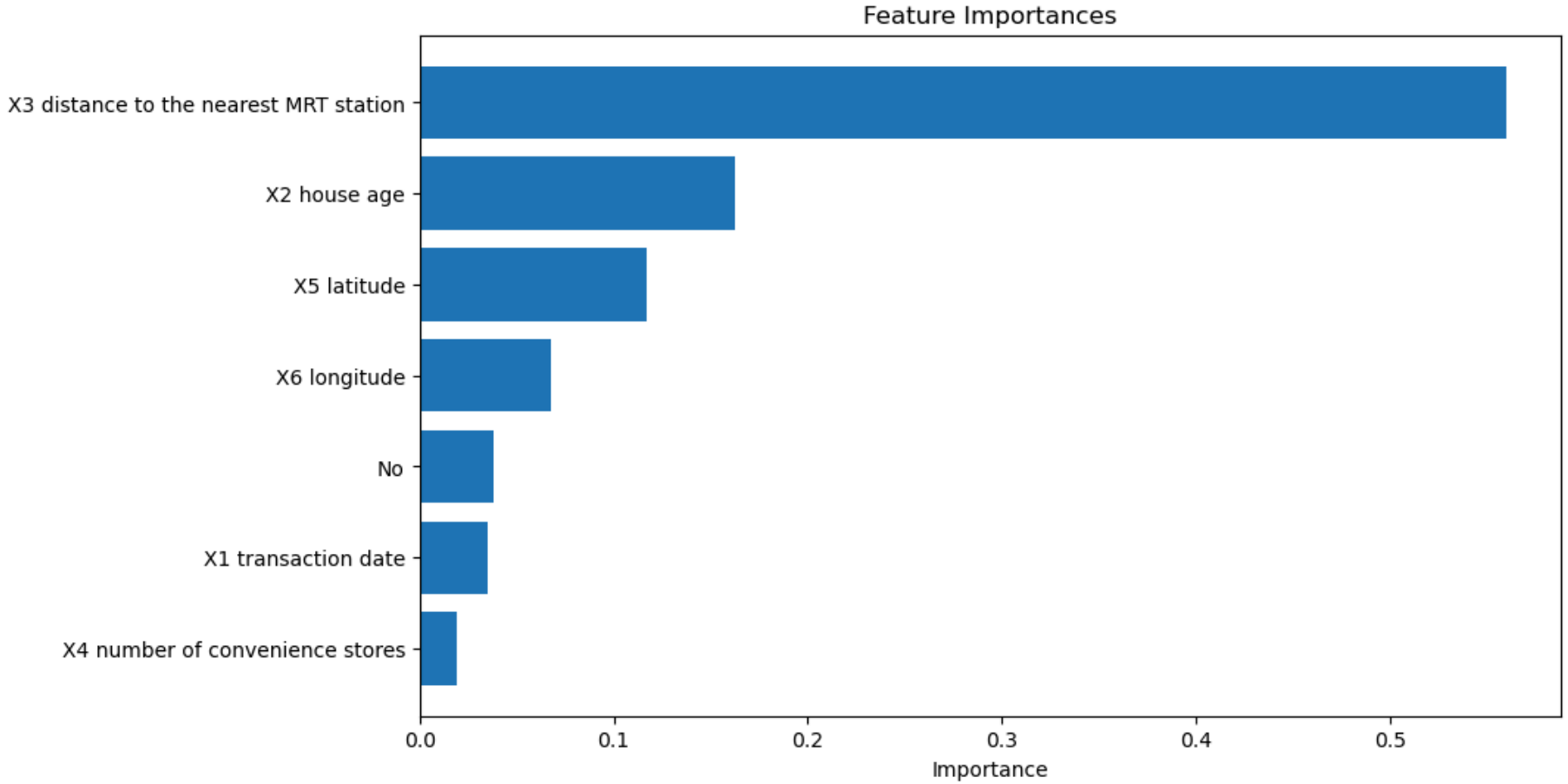
In [12]:
```
# Make predictions
y_pred = rf_model.predict(X_test)
```

In [13]:
```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"R-squared (R2): {r2:.2f}")
```

```
Mean Squared Error (MSE): 31.85
R-squared (R2): 0.81
```

In [14]:
```
# Feature importance
importances = rf_model.feature_importances_
feature_names = features.columns
indices = importances.argsort()
```

In [15]:
```
plt.figure(figsize=(10, 6))
plt.title("Feature Importances")
plt.barh(range(len(indices)), importances[indices], align="center")
plt.yticks(range(len(indices)), [feature_names[i] for i in indices])
plt.xlabel("Importance")
plt.show()
```



In [16]:
```
# Scatter plot of actual vs. predicted values
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.5, color='blue', label='Predicted vs Actual')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], 'r--', lw=2, label='Perfect Prediction')
plt.xlabel('Actual House Prices')
plt.ylabel('Predicted House Prices')
plt.title('Actual vs. Predicted House Prices')
plt.legend()
plt.show()
```


Actual vs. Predicted House Prices
Predicted vs Actual
Perfect Prediction
Predicted House Prices
Actual House Prices
10
20
30
40
50
60

In [ ]: