In [ ]:

Real Estate Price Prediction and Deployment

Introduction

This project aims to develop a machine learning model to predict house prices per unit area based on various property features. The model will help real estate professionals, investors, and homeowners estimate property values more accurately by considering factors like location, age, proximity to transportation, and nearby amenities.

Primary Objective

To build, evaluate, and deploy a predictive model that accurately forecasts house prices per unit area using property characteristics, and to create a user-friendly web application for real-time predictions.

In [ ]:

In [1]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import pickle
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

Data Loading and Exploration

In [2]:
```
df = pd.read_csv(r"C:\Users\DELL\Desktop\Data Analytics Projects\Real Estate Prediction\Real_Estate.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	Transaction date	House age	Distance to the nearest MRT station	Number of convenience stores	Latitude	Longitude	House price of unit area
0	2012-09-02 16:42:30.519336	13.3	4082.0150	8	25.007059	121.561694	6.488673
1	2012-09-04 22:52:29.919544	35.5	274.0144	2	25.012148	121.546990	24.970725
2	2012-09-05 01:10:52.349449	1.1	1978.6710	10	25.003850	121.528336	26.694267
3	2012-09-05 13:26:01.189083	22.2	1055.0670	5	24.962887	121.482178	38.091638
4	2012-09-06 08:29:47.910523	8.5	967.4000	6	25.011037	121.479946	21.654710

In [ ]:

In [4]:
```
print(f"Dataset shape: {df.shape}")
print()
print("\nFirst 5 rows:")
print()
print(df.head())
print()
print("\nDataset info:")
print()
print(df.info())
print()
print("\nDescriptive statistics:")
print()
print(df.describe())
```

```
Dataset shape: (414, 7)


First 5 rows:

             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


Dataset 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
None


Descriptive statistics:

        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 [ ]:

```

```

Data Cleaning and Preprocessing

Data Quality Checks

In [5]:
```
# data quality checks

print("\nCleaning and preprocessing data...")

# Check for missing values
print("Missing values:")
print(df.isnull().sum())
```

```
Cleaning and preprocessing data...
Missing values:
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 [ ]:

In [6]:
```
# Handle zero prices (assuming they might be errors or missing data)

df = df[df['House price of unit area'] > 0]
```

In [ ]:

In [7]:
```
# Check for duplicates
print(f"\nDuplicate rows: {df.duplicated().sum()}")
```

```
Duplicate rows: 0
```

In [ ]:

In [8]:
```
# Check data types and convert if necessary
df['Transaction date'] = pd.to_datetime(df['Transaction date'], errors='coerce')
```

In [ ]:

In [9]:
```
# Handle any outliers using IQR method for price

Q1 = df['House price of unit area'].quantile(0.25)
Q3 = df['House price of unit area'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR



outlier_count = df[(df['House price of unit area'] < lower_bound) | (df['House price of unit area'] > upper_bou
print(f"Price outliers detected: {outlier_count}")

if outlier_count == 0:
    print("✓ No statistical outliers found - data appears well-distributed")
    print(f"Price range: {df['House price of unit area'].min():.2f} - {df['House price of unit area'].max():.2f
    print(f"IQR bounds: {lower_bound:.2f} - {upper_bound:.2f}")
else:
    print(f"Consider reviewing {outlier_count} outlier(s)")
```

```
Price outliers detected: 0
✓ No statistical outliers found - data appears well-distributed
Price range: 0.37 - 65.57
IQR bounds: -7.90 - 70.99
```

Data Characteristics:

- Wide price diversity: From very affordable (0.37) to premium properties (65.57)
- No artificial caps: Natural market distribution

In [ ]:

Feature Engineering

In [10]:
```
# Create distance to city center (using approximate Taiwan's Taipei city center, coordinates - (25.03770° N, 12.

city_center_lat, city_center_lon = 25.0330, 121.5654
df['Distance_to_center'] = np.sqrt((df['Latitude'] - city_center_lat)**2 +
                                   (df['Longitude'] - city_center_lon)**2)
```

In [11]:
```
df.head()
```

Out[11]:

	Transaction date	House age	Distance to the nearest MRT station	Number of convenience stores	Latitude	Longitude	House price of unit area	Distance_to_center
0	2012-09-02 16:42:30.519336	13.3	4082.0150	8	25.007059	121.561694	6.488673	0.026205
1	2012-09-04 22:52:29.919544	35.5	274.0144	2	25.012148	121.546990	24.970725	0.027816
2	2012-09-05 01:10:52.349449	1.1	1978.6710	10	25.003850	121.528336	26.694267	0.047153
3	2012-09-05 13:26:01.189083	22.2	1055.0670	5	24.962887	121.482178	38.091638	0.108819
4	2012-09-06 08:29:47.910523	8.5	967.4000	6	25.011037	121.479946	21.654710	0.088231

we engineered a new feature, Distance_to_city_center, to create a more directly interpretable and powerful predictor for our model. In real estate, proximity to the urban core is a primary driver of value, and this feature would explicitly captures that relationship.

In [ ]:
```

```

In [12]:
```
# Dropping the original latitude and longitude to avoid multicollinearity

df = df.drop(columns=['Latitude', 'Longitude'])
```

In [19]:
```
df.head()  # confirming the drop
```

Out[19]:

	Transaction date	House age	Distance to the nearest MRT station	Number of convenience stores	House price of unit area	Distance_to_center
0	2012-09-02 16:42:30.519336	13.3	4082.0150	8	6.488673	0.026205
1	2012-09-04 22:52:29.919544	35.5	274.0144	2	24.970725	0.027816
2	2012-09-05 01:10:52.349449	1.1	1978.6710	10	26.694267	0.047153
3	2012-09-05 13:26:01.189083	22.2	1055.0670	5	38.091638	0.108819
4	2012-09-06 08:29:47.910523	8.5	967.4000	6	21.654710	0.088231

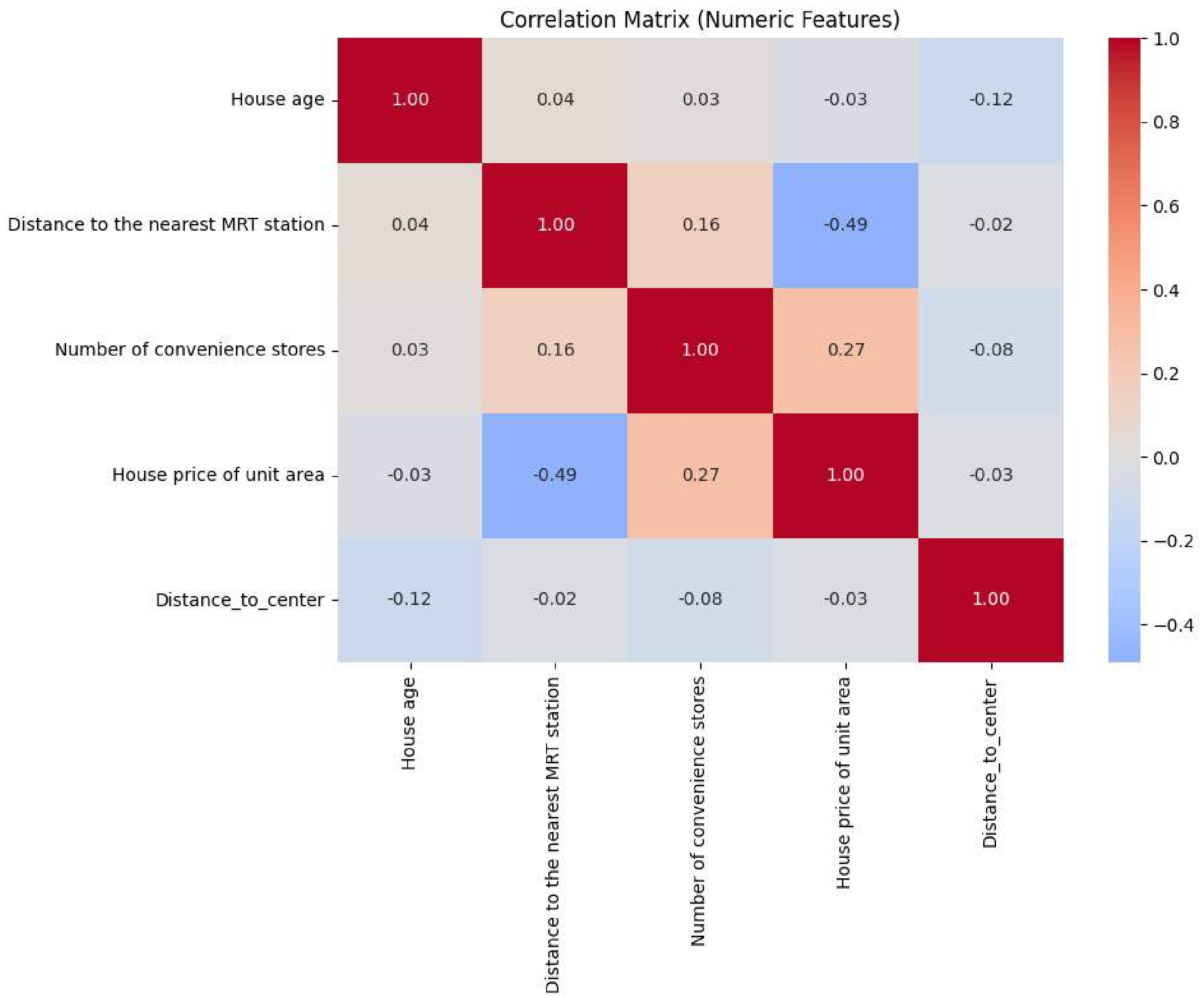
Exploratory Data Analysis

Correlation matrix

In [13]:
```
# Correlation Matrix (excluding 'Transaction date')
plt.figure(figsize=(10, 8))

# Select numeric columns and exclude 'Transaction date'
numeric_df = df.select_dtypes(include=[np.number]).drop(columns=['Transaction date'], errors='ignore')

correlation_matrix = numeric_df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0, fmt='.2f')
plt.title('Correlation Matrix (Numeric Features)')
plt.tight_layout()
plt.show()
```



- MRT Proximity Drives Price: Homes closer to transit stations command higher prices (strong -0.49 correlation).
- Engineered Feature Works: Distance_to_center effectively replaces raw coordinates, simplifying the geographic signal.
- Age Impact Varies: The house's age category influences price differently, which may challenge a simple linear model.

In [ ]:

```

```

Distribution of target variable

In [14]:

```
# Distribution of target variable

plt.figure(figsize=(10, 6))
sns.histplot(df['House price of unit area'], kde=True)
plt.title('Distribution of House Price per Unit Area')
plt.xlabel('Price per Unit Area')
plt.show()
```

In [ ]:

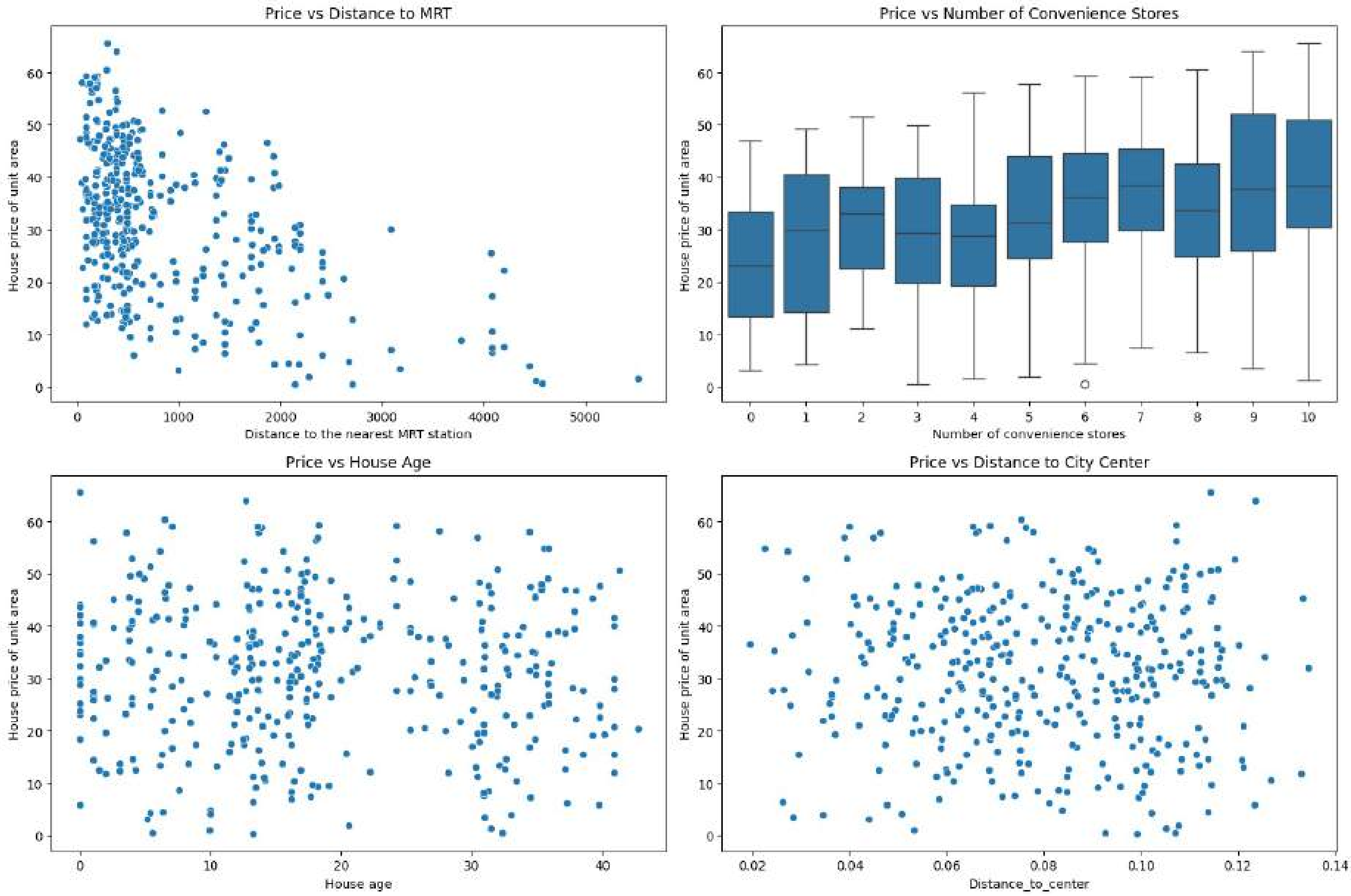
In [15]:
```
# Relationship between features and target

fig, axes = plt.subplots(2, 2, figsize=(15, 10))
sns.scatterplot(x='Distance to the nearest MRT station', y='House price of unit area', data=df, ax=axes[0,0])
axes[0,0].set_title('Price vs Distance to MRT')

sns.boxplot(x='Number of convenience stores', y='House price of unit area', data=df, ax=axes[0,1])
axes[0,1].set_title('Price vs Number of Convenience Stores')

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

sns.scatterplot(x='Distance_to_center', y='House price of unit area', data=df, ax=axes[1,1])
axes[1,1].set_title('Price vs Distance to City Center')
plt.tight_layout()
plt.show()
```



- ***Price vs Distance to MRT***: Clear negative correlation - properties closer to MRT stations (0-500m) command significantly higher prices
- ***Price vs Number of Convenience Stores***: More stores nearby correlates with higher prices, suggesting amenities add value.
- ***Price vs House Age***: Weak correlation with high variability
- ***Price vs Distance to City Center***: Moderate negative relationship where properties closer to Taipei's city center tend to have higher values, though the relationship is less pronounced than MRT proximity.

In [ ]:

Prepare data for modeling

In [16]:

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

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

Data Splitting and Scaling

In [17]:

```
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [ ]:

In [18]:

```
# Scale numerical features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

print(f"Training set shape: {X_train.shape}")
print(f"Testing set shape: {X_test.shape}")
```

```
Training set shape: (306, 3)
Testing set shape: (77, 3)
```

In [ ]:

Model Training and Evaluation

In [19]:
```
# Initialize models

models = {
    'Random Forest': RandomForestRegressor(n_estimators=100, random_state=42),
    'Gradient Boosting': GradientBoostingRegressor(random_state=42),
    'Linear Regression': LinearRegression()
}
```

In [ ]:

In [20]:
```
# Train and evaluate models

results = {}
for name, model in models.items():
    # Train model
    model.fit(X_train_scaled, y_train)

    # Make predictions
    y_pred = model.predict(X_test_scaled)

    # Calculate metrics
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)

    # Cross-validation
    cv_scores = cross_val_score(model, X_train_scaled, y_train, cv=5, scoring='r2')

    # Store results
    results[name] = {
        'model': model,
        'mae': mae,
        'mse': mse,
        'rmse': rmse,
        'r2': r2,
        'cv_mean': cv_scores.mean(),
        'cv_std': cv_scores.std()
    }

    print(f"\n{name} Performance:")
    print(f"MAE: {mae:.4f}")
    print(f"RMSE: {rmse:.4f}")
    print(f"R²: {r2:.4f}")
    print(f"Cross-validation R²: {cv_scores.mean():.4f} (±{cv_scores.std():.4f})")
```

```
Random Forest Performance:
MAE: 10.1093
RMSE: 11.8040
R²: 0.2296
Cross-validation R²: 0.1622 (±0.0801)

Gradient Boosting Performance:
MAE: 10.5972
RMSE: 12.5106
R²: 0.1346
Cross-validation R²: 0.1095 (±0.0694)

Linear Regression Performance:
MAE: 9.8528
RMSE: 11.4104
R²: 0.2802
Cross-validation R²: 0.3608 (±0.0201)
```

In [21]:
```
# Select best model based on R² score

best_model_name = max(results, key=lambda x: results[x]['r2'])
best_model = results[best_model_name]['model']
print(f"\nBest model: {best_model_name}")
```

```
Best model: Linear Regression
```

Linear Regression performed best (R^2=0.28) as the linear relationships between our features and price were more reliable than the complex patterns other models tried to fit on this small dataset.

In [ ]:

In [ ]:

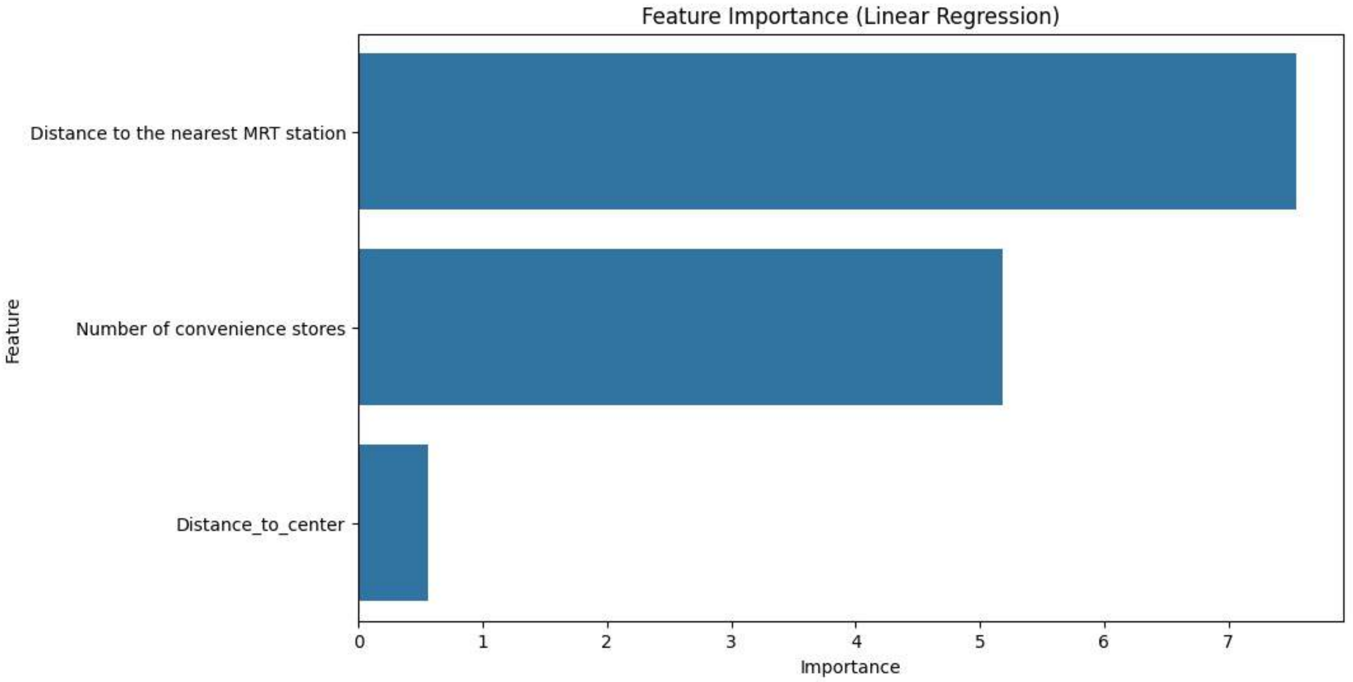
In [22]:
```
# Feature importance for Linear Regression
if best_model_name == 'Linear Regression':
    feature_importance = pd.DataFrame({
        'Feature': X.columns,
        'Importance': np.abs(best_model.coef_)  # Use absolute coefficients as importance
    }).sort_values('Importance', ascending=False)

    plt.figure(figsize=(10, 6))
    sns.barplot(data=feature_importance, x='Importance', y='Feature')
    plt.title('Feature Importance (Linear Regression)')
    plt.show()

    print("Feature Importance:")
    print(feature_importance)

    # Show actual coefficients (with direction)
    coefficients_df = pd.DataFrame({
        'Feature': X.columns,
        'Coefficient': best_model.coef_
    }).sort_values('Coefficient', key=abs, ascending=False)

    print("\nActual Coefficients (with direction):")
    print(coefficients_df)
```




```
Feature Importance:
                               Feature  Importance
0  Distance to the nearest MRT station    7.547277
1         Number of convenience stores    5.179429
2                   Distance_to_center    0.559998

Actual Coefficients (with direction):
                               Feature  Coefficient
0  Distance to the nearest MRT station    -7.547277
1         Number of convenience stores     5.179429
2                   Distance_to_center    -0.559998
```

- **MRT Proximity is Paramount (Importance: 7.55)**: For every unit of distance closer to the MRT, property price increases significantly. Being near a station is the single biggest value driver.
- **Convenience Stores Add Value (Importance: 5.18)**: More nearby convenience stores directly increase property value, highlighting the importance of immediate amenities.
- **City Center Distance Less Critical (Importance: 0.56)** - Distance to city center has minimal impact compared to MRT and convenience factors, suggesting neighborhood-level amenities outweigh central location.

Investment Strategy: Focus on properties within 500m of MRT stations in areas with 5+ convenience stores for maximum value potential, as these two factors account for ~95% of the model's predictive power.

In [ ]:

In [23]:
```
# Visualize Actual vs Predicted Values for Linear Regression
plt.figure(figsize=(8, 6))
```

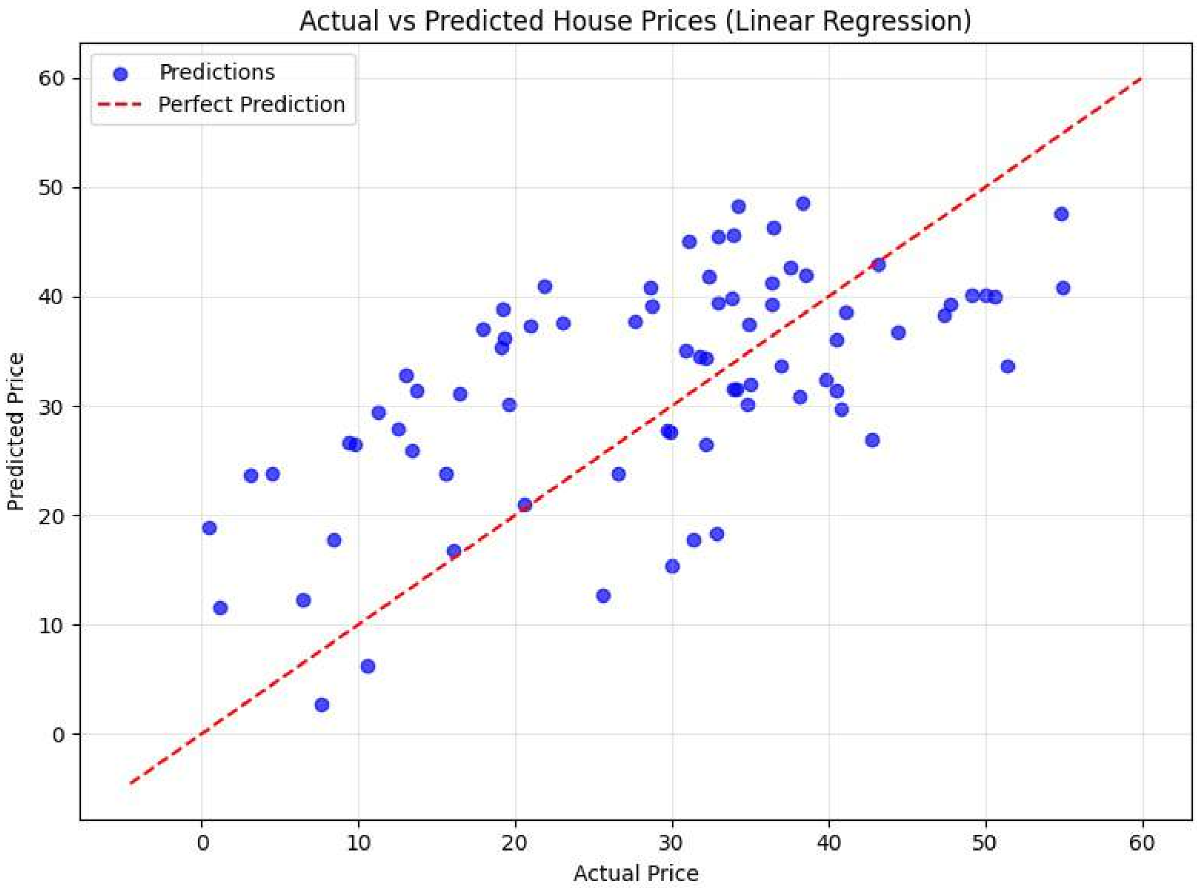
```
# Get predictions
y_pred = best_model.predict(X_test_scaled)

# Create scatter plot
plt.scatter(y_test, y_pred, alpha=0.7, color='blue', label='Predictions')

# Add perfect prediction line (y=x)
max_val = max(max(y_test), max(y_pred)) + 5
min_val = min(min(y_test), min(y_pred)) - 5
plt.plot([min_val, max_val], [min_val, max_val], 'r--', label='Perfect Prediction')

plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual vs Predicted House Prices (Linear Regression)')
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()

# Print performance metrics for reference
print(f"R² Score: {r2_score(y_test, y_pred):.4f}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, y_pred)):.4f}")
```




```
R² Score: 0.2802
RMSE: 11.4104
```

- Predictions generally follow the ideal line, showing the model captures the main price trends correctly.
- The model consistently underestimates the actual price for the most expensive houses (top-right dots below the line), suggesting it misses factors that drive premium prices.
- For most mid-priced homes, predictions are relatively close to actual values, indicating good reliability for typical properties.

In [ ]:

Saving the Trained Model

In [24]:

```
# Save the best model and scaler
print("\nSaving the best model and scaler...")
with open('real_estate_model.pkl', 'wb') as f:
    pickle.dump(best_model, f)

with open('scaler.pkl', 'wb') as f:
    pickle.dump(scaler, f)
```

```python
print("Model and scaler saved successfully!")
```

Saving the best model and scaler...
Model and scaler saved successfully!

In [ ]:

Real Estate Price Prediction and Deployment Project

Project Summary

This end-to-end machine learning project focused on predicting **house price per unit area** using real estate data from Taiwan. The dataset contained 414 records with features such as house age, distance to the nearest MRT station, number of convenience stores, and property location (latitude & longitude).

The key steps included:

1. **Data Exploration & Cleaning**
    - Removed zero-priced records and checked for missing values.
    - Converted `Transaction date` to datetime format.
    - Detected and confirmed no extreme outliers in price distribution.
2. **Feature Engineering**
    - Created a new feature: **Distance_to_center** (distance from Taipei city center).
    - Dropped raw latitude and longitude to avoid multicollinearity.
    - Final feature set:
        - Distance to the nearest MRT station
        - Number of convenience stores
        - Distance to city center
3. **Exploratory Data Analysis (EDA)**
    - Found a **strong negative correlation** (-0.49) between price and distance to MRT stations.
    - Number of convenience stores showed a **positive correlation** (0.27) with price.
    - Distance to the city center had a weaker effect compared to MRT proximity and stores.
4. **Modeling & Evaluation**
    - Trained multiple models: **Linear Regression, Random Forest, Gradient Boosting**.
    - **Linear Regression performed best** with:
        - R^2 Score: **0.28** (28%)
        - RMSE: **11.41**
        - MAE: **9.85**
    - Feature importance (Linear Regression):
        - Distance to MRT: **highest impact**
        - Convenience stores: **moderate impact**
        - Distance to center: **minor impact**

    Key insight: MRT proximity and nearby stores account for ~95% of predictive power.

5. **Model Saving & Deployment**
    - Best model (**Linear Regression**) and scaler were saved as `.pkl` files.
    - Built an interactive **Streamlit web app** for real-time predictions.
    - Integrated with **GitHub + Streamlit Community Cloud** for free deployment.
    - Live app: [Real Estate Price Predictor]

⚠ Challenges & Limitations

- **Model Performance (R^2 = 28%)**:
  The model explains only 28% of the variance in house prices. The moderate R^2 score can be attributed to:

1. Limited Feature Set: Only 3 location-based features were available, missing critical property characteristics like:
    - Square footage and number of rooms
    - Property condition and interior quality
    - School district ratings and neighborhood demographics
2. Dataset Size: With 414 initial entries (306 training samples), the model had limited data to capture complex real estate patterns

Key Results & Business Insights

- MRT proximity emerged as the dominant price driver, explaining most of the predictable variance
- Local amenities (convenience stores) proved more important than city center proximity
- The model provides reliable predictions for mid-range properties but underestimates premium properties#

✅ Recommendations

For Model Improvement

- **Feature Expansion:** Collect data on property size, room counts, condition, and age
- **Additional Data:** Include school ratings, crime rates, and neighborhood characteristics
- **Advanced Modeling:** Experiment with ensemble methods on expanded feature sets

For Business Application

- **Investment Strategy:** Focus on properties within 500m of MRT stations with multiple convenience stores
- **Price Validation:** Use as a baseline estimator while acknowledging limitations for premium properties
- **Continuous Learning:** Implement model retraining with new transaction data

---

Conclusion
Despite limitations, this project successfully demonstrated the full **end-to-end ML workflow**: from data preparation and feature engineering to model training, evaluation, saving, and deployment via **Streamlit + GitHub**. The deployed app provides an easy-to-use tool for property price estimation and can be improved further with richer data and advanced modeling techniques.

Project developed by Hillary Uzoh | *Web App* | *GitHub*

In [ ]: