Housing Price Prediction Using Machine Learning

Project Overview

Accurate housing price prediction is essential for **real estate companies, investors, and buyers** to make informed decisions.

In this project, I built and evaluated **machine learning regression models** to predict **house prices based on property features** such as size, rooms, location attributes, and amenities.

The project focuses on:

- Data preparation & feature engineering
- Regression model development
- Performance evaluation and comparison

Business Problem

Can we accurately predict house prices based on property characteristics?

This problem is relevant for:

- Real estate pricing strategies
- Market valuation tools
- Investment analysis

Dataset Description

The housing dataset is used is called “Housing Prices Dataset” by ‘M Yasser H’ (Link).

The dataset includes the following features:

Categorical Features

- Main road access
- Guest room
- Basement
- Hot water heating
- Air conditioning
- Preferred area
- Furnishing status (furnished / semi-furnished / unfurnished)

Numerical Features

- Price (target variable)
- Area
- Bedrooms
- Bathrooms
- Stories
- Parking spaces

Figure 1: The Dataset

```
import pandas as pd
df=pd.read_csv('/content/archive (1).zip')
df
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	13300000	7420	4	2	3	yes	no	no	no	yes	2	yes	furnished
1	12250000	8960	4	4	4	yes	no	no	no	yes	3	no	furnished
2	12250000	9960	3	2	2	yes	no	yes	no	no	2	yes	semi-furnished
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes	furnished
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no	furnished
...	...	...	...	...	...	...	...	...	...	...	...	...	...
540	1820000	3000	2	1	1	yes	no	yes	no	no	2	no	unfurnished
541	1767150	2400	3	1	1	no	no	no	no	no	0	no	semi-furnished
542	1750000	3620	2	1	1	yes	no	no	no	no	0	no	unfurnished
543	1750000	2910	3	1	1	no	no	no	no	no	0	no	furnished
544	1750000	3850	3	1	2	yes	no	no	no	no	0	no	unfurnished

545 rows × 13 columns

Data Preparation & Cleaning

Steps Performed:

- Loaded dataset into Python (Pandas)
- Checked for:
    - Missing values - **None found**
    - Duplicate records - **None found**
- Verified data types and feature consistency

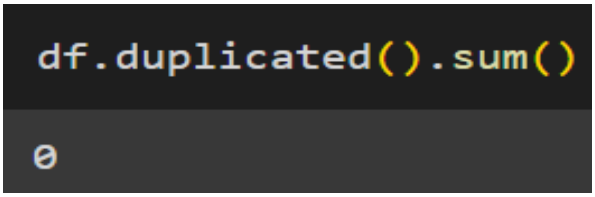
Outcome: Dataset required **no imputation or row removal**, making it suitable for modeling.

Figure 2: Fining Missing and Null Values

```
df.isnull().sum()
```

	0
price	0
area	0
bedrooms	0
bathrooms	0
stories	0
mainroad	0
guestroom	0
basement	0
hotwaterheating	0
airconditioning	0
parking	0
prefarea	0
furnishingstatus	0

Figure 3: Looking for Any Duplicates



Feature Encoding & Transformation

Machine learning models require numerical input, so categorical features were encoded:

- Binary categorical features (Yes/No) → **0 / 1**
- Furnishing status → **Label Encoding**
    - Furnished → 0
    - Semi-furnished → 1
    - Unfurnished → 2

Why this approach

Label encoding preserved dataset size and enabled efficient model training without unnecessary feature expansion.

Figure 3: Transforming Data using Mapping and Label Encoding Methods

```python
import pandas as pd
from sklearn.preprocessing import LabelEncoder

columns_to_transform = ['mainroad', 'guestroom', 'basement',
                        'hotwaterheating', 'airconditioning', 'prefarea']

for column in columns_to_transform:
    df[column] = df[column].map({'yes': 1, 'no': 0})

label_encoder = LabelEncoder()

df['furnishingstatus_encoded'] = label_encoder.fit_transform(df['furnishingstatus'])

print(df.head())
```

```
      price  area  bedrooms  bathrooms  stories  mainroad  guestroom
0  13300000  7420         4          2        3         1          0
1  12250000  8960         4          4        4         1          0
2  12250000  9960         3          2        2         1          0
3  12215000  7500         4          2        2         1          0
4  11410000  7420         4          1        2         1          1

   basement  hotwaterheating  airconditioning  parking  prefarea  \
0         0                0                1        2         1
1         0                0                1        3         0
2         1                0                0        2         1
3         1                0                1        3         1
4         1                0                1        2         0

  furnishingstatus  furnishingstatus_encoded
0        furnished                         0
1        furnished                         0
2   semi-furnished                         1
3        furnished                         0
4        furnished                         0
```

Exploratory Data Analysis (EDA)

EDA was performed to understand distributions, relationships, and potential anomalies.

Data Distribution Visualizations:

Figure 4: Data Distribution Visualization using histogram

```
import matplotlib.pyplot as plt

numerical_features = ['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking']

fig, axes = plt.subplots(2, 3, figsize=(15, 8))
fig.suptitle('Distribution of Numerical Features in Housing Data', fontsize=16)

axes = axes.ravel()

for i, feature in enumerate(numerical_features):
    axes[i].hist(df[feature], bins=30, color='skyblue', edgecolor='black')
    axes[i].set_title(f'{feature.capitalize()}')
    axes[i].set_xlabel(feature)
    axes[i].set_ylabel('Frequency')

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

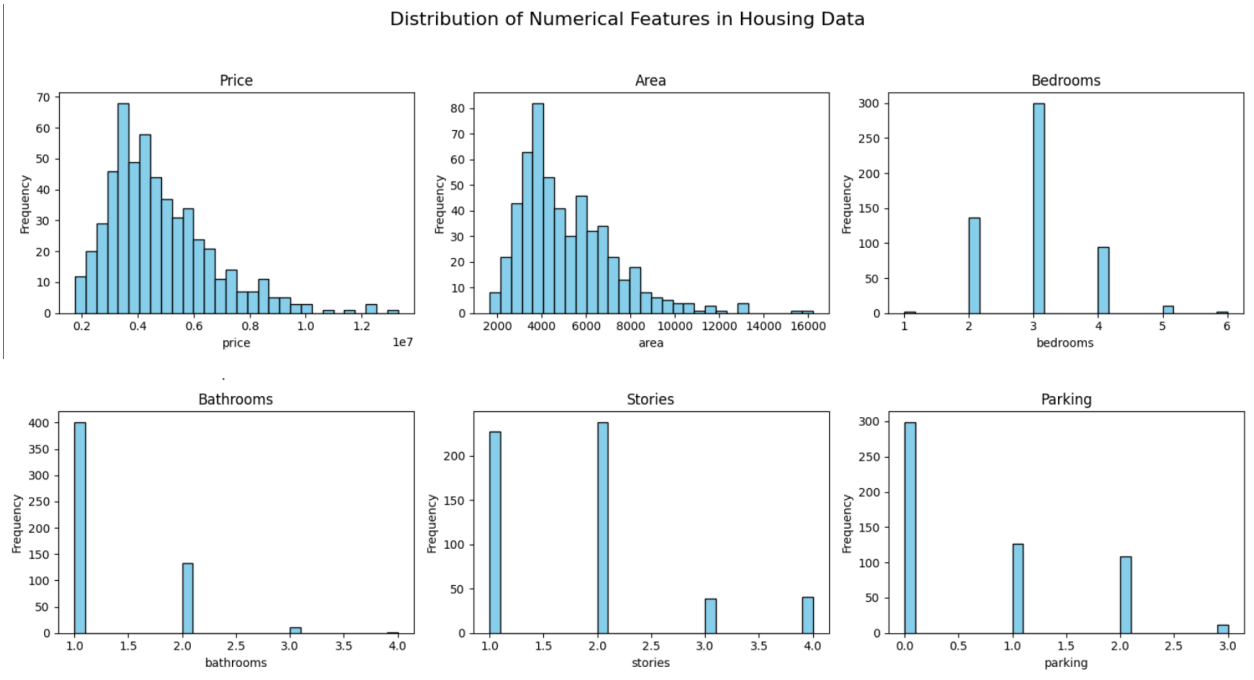
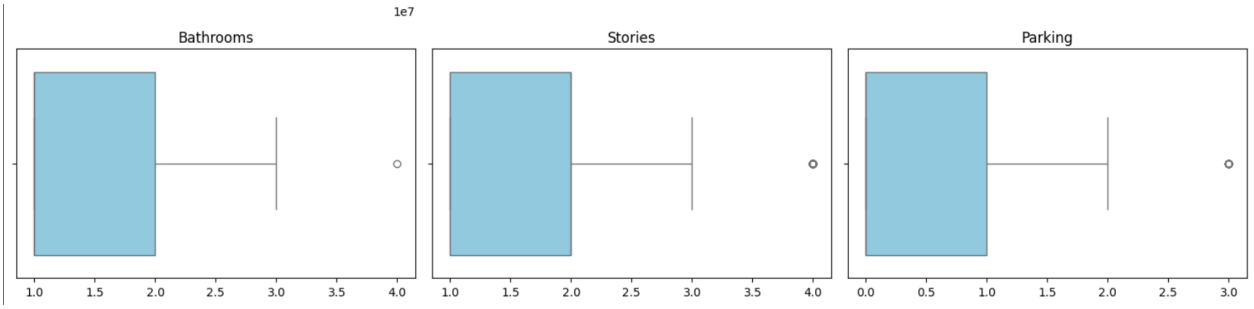
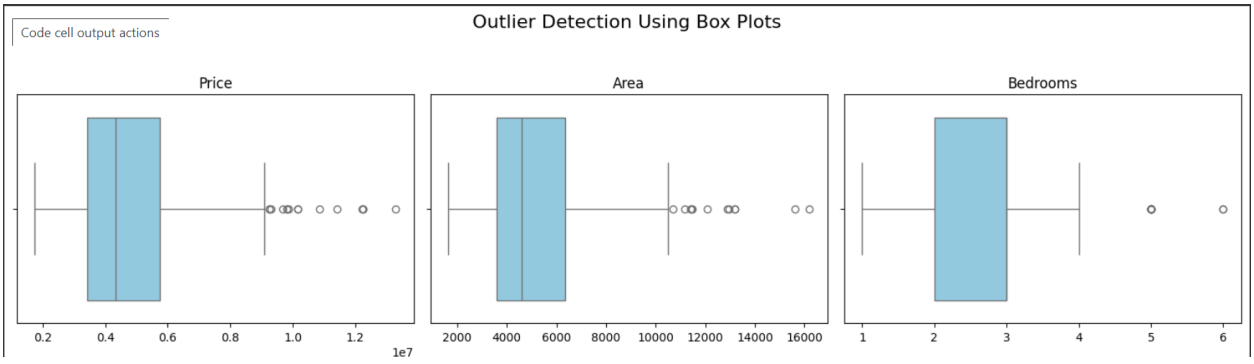


Figure 5: Identifying Outliers using Box Plot Visualization

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

numerical_features = ['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking']

fig, axes = plt.subplots(2, 3, figsize=(15, 8))
fig.suptitle('Outlier Detection Using Box Plots', fontsize=16)

axes = axes.ravel()

for i, feature in enumerate(numerical_features):
    sns.boxplot(data=df, x=feature, ax=axes[i], color='skyblue')
    axes[i].set_title(f'{feature.capitalize()}')
    axes[i].set_xlabel('')

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





Key Observations:

- Price increases with area and number of rooms
- Outliers exist but represent valid extreme properties
- Strong positive correlation between area and price

Machine Learning Approach

Since the target variable (**price**) is continuous, this is a **supervised regression problem**.

Models Implemented:

1. **Linear Regression**
    - Captures linear relationships
    - Simple, interpretable baseline model
2. **Random Forest Regression**
    - Captures non-linear relationships
    - Ensemble-based approach

Train-Test Split:

- 80% training
- 20% testing
- Random state set for reproducibility

Model 1: Linear Regression

Figure 6: Importing, Fitting, Testing and Training Liner Regression Model

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

feature_columns = ['area', 'bedrooms', 'bathrooms', 'stories', 'mainroad', 'guestroom',
                   'basement', 'hotwaterheating', 'airconditioning', 'parking', 'prefarea',
                   'furnishingstatus_encoded']
target_column = 'price'

X = df[feature_columns]
y = df[target_column]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

Evaluation Metrics:

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- R^2 Score

Figure 7: Evaluating Linear Regression Model using MAE, MSE, RMSE, R2-Score

```python
# Evaluate the model
print("Mean Absolute Error (MAE):", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error (RMSE):", mean_squared_error(y_test, y_pred, squared=False))
print("R^2 Score:", r2_score(y_test, y_pred))
```

```
Mean Absolute Error (MAE): 979679.6912959901
Mean Squared Error (MSE): 1771751116594.0352
Root Mean Squared Error (RMSE): 1331071.4167895108
R^2 Score: 0.6494754192267803
```

Evaluation Results:

- MAE: 979,679
- RMSE: 1,331,071
- R^2 Score: 0.64

Interpretation:

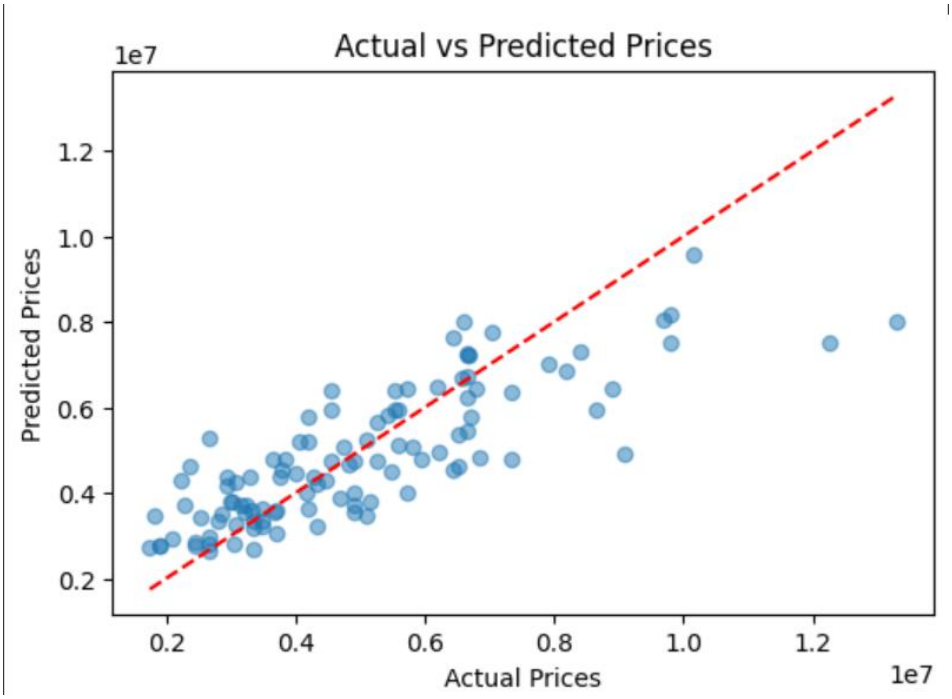
The model explains ~64% of price variance — a solid baseline for structured housing data.

Linear Regression Visualizations

- Area vs Price regression plot
- Residual plot
- Actual vs Predicted prices

Figure 8: Visualizing Actual Vs Predicted Data Points

```python
# Visualization 3: Actual vs Predicted Scatter Plot
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--')
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Actual vs Predicted Prices')
plt.show()
```



Insight: Most the plots are closer to the red line indicating good performing model.

Model 2: Random Forest Regression

Random Forest was used to capture **non-linear interactions** between features.

Figure 9: Importing, Fitting, Testing and Training Random Forest Regression Model

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

feature_columns = ['area', 'bedrooms', 'bathrooms', 'stories', 'mainroad', 'guestroom',
                   'basement', 'hotwaterheating', 'airconditioning', 'parking', 'prefarea',
                   'furnishingstatus_encoded']
target_column = 'price'

X = df[feature_columns]
y = df[target_column]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

Model Evaluation:

Figure 10: Evaluating Random Forest Model using MAE, MSE, RMSE, R2-Score

```
print("Mean Absolute Error (MAE):", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error (RMSE):", mean_squared_error(y_test, y_pred, squared=False))
print("R^2 Score:", r2_score(y_test, y_pred))
```

```
Mean Absolute Error (MAE): 1025289.6821100918
Mean Squared Error (MSE): 1963538216518.6526
Root Mean Squared Error (RMSE): 1401263.0789821919
R^2 Score: 0.6115321143409216
```

Evaluation Results:

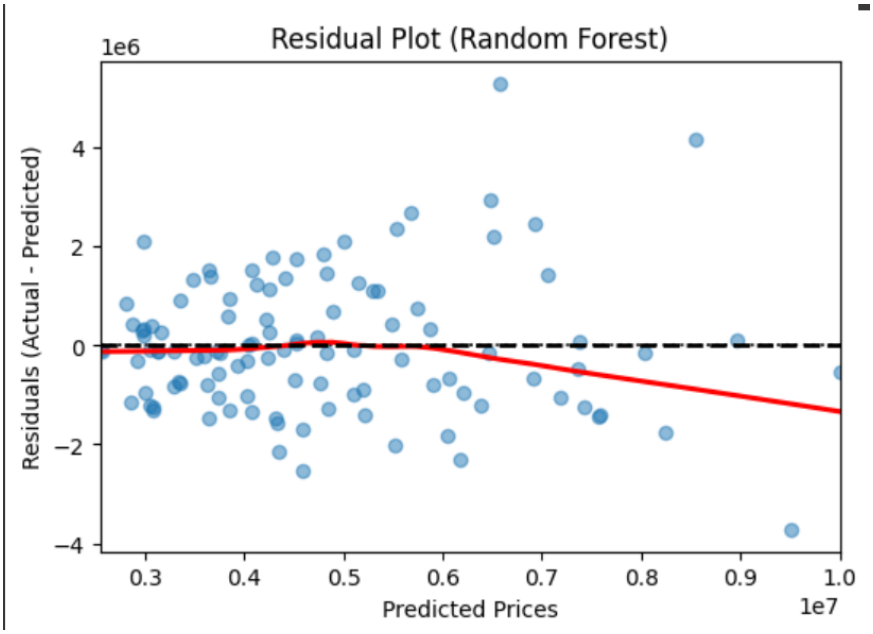
- MAE: 1,025,289
- RMSE: 1,401,263
- R^2 Score: 0.61

Random Forest Model Visualizations

Figure 11: Using Residual Plot Visualizing Actual Vs Predicted Data Points

```
# Visualization 2: Residual Plot
residuals = y_test - y_pred
plt.figure(figsize=(8, 6))
sns.residplot(x=y_pred, y=residuals, lowess=True, line_kws={"color": "red"}, scatter_kws={"alpha": 0.5})
plt.xlabel('Predicted Prices')
plt.ylabel('Residuals (Actual - Pred
plt.title('Residual Plot (Random Forest)')
plt.axhline(y=0, color='black', linestyle='--', linewidth=2)
plt.show()
```

Loading...



Insight: While powerful, the Random Forest model did **not outperform Linear Regression** on this dataset.

Model Comparison

Linear Regression Model

- MAE: 979,679
- RMSE: 1,331,071
- R^2 Score: **0.64**

Linear Regression Model

- MAE: 1,025,289
- RMSE: 1,401,263
- R^2 Score: **0.61**

Final Recommendation

For this dataset:

- **Linear Regression** is the preferred model
- It provides:
    - Better performance
    - Simpler interpretation
    - Faster training
    - Easier deployment

Link to the “Housing Prices Dataset” by ‘M Yasser H’:

https://www.kaggle.com/datasets/yasserh/housing-prices-dataset/data