Data Analysis and Predictive Modeling of House Prices in India

This presentation explores key factors influencing house prices, using data visualization and machine learning models to predict prices accurately within the Indian real estate market.

by Rishabh Singh

Dataset Overview

Source

The dataset "House Price India.csv" was used for this analysis which was sourced from kaggle.com .

Linkhttps://www.kaggle.com/datasets/mohamedafsal007/house-price-dataset-of-india/data

The Data consist of total 14621 rows and 23 columns

Initial Analysis

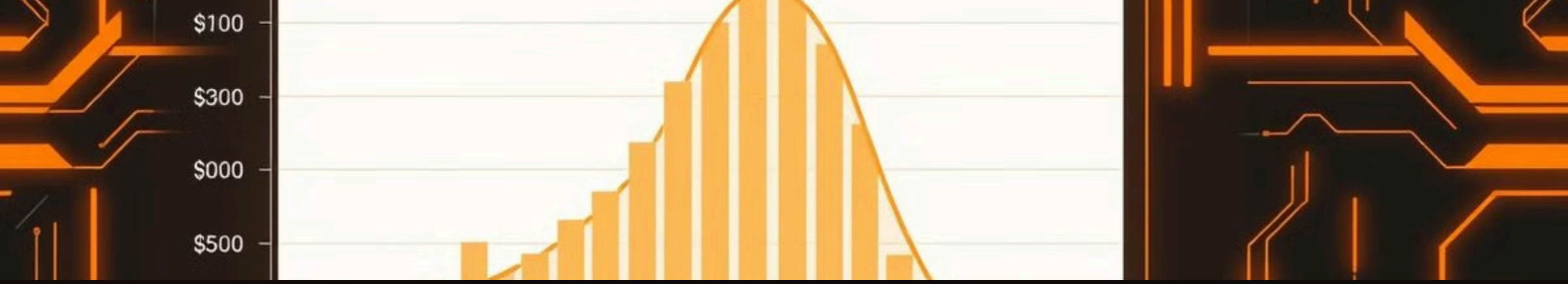
Missing values and duplicate records were identified and addressed.

```
[ ] house.duplicated().sum() #checking for the duplicates

    0
```

```
house.isnull().sum() #checking for null values
```

	0
id	0
Date	0
number of bedrooms	0
number of bathrooms	0
living area	0
lot area	0
number of floors	0
waterfront present	0
number of views	0
condition of the house	0
grade of the house	0
Area of the house(excluding basement)	0
Area of the basement	0
Built Year	0
Renovation Year	0
Postal Code	0
Lattitude	0
Longitude	0
living_area_renov	0
lot_area_renov	0
Number of schools nearby	0
Distance from the airport	0
Price	0

dtype: int64



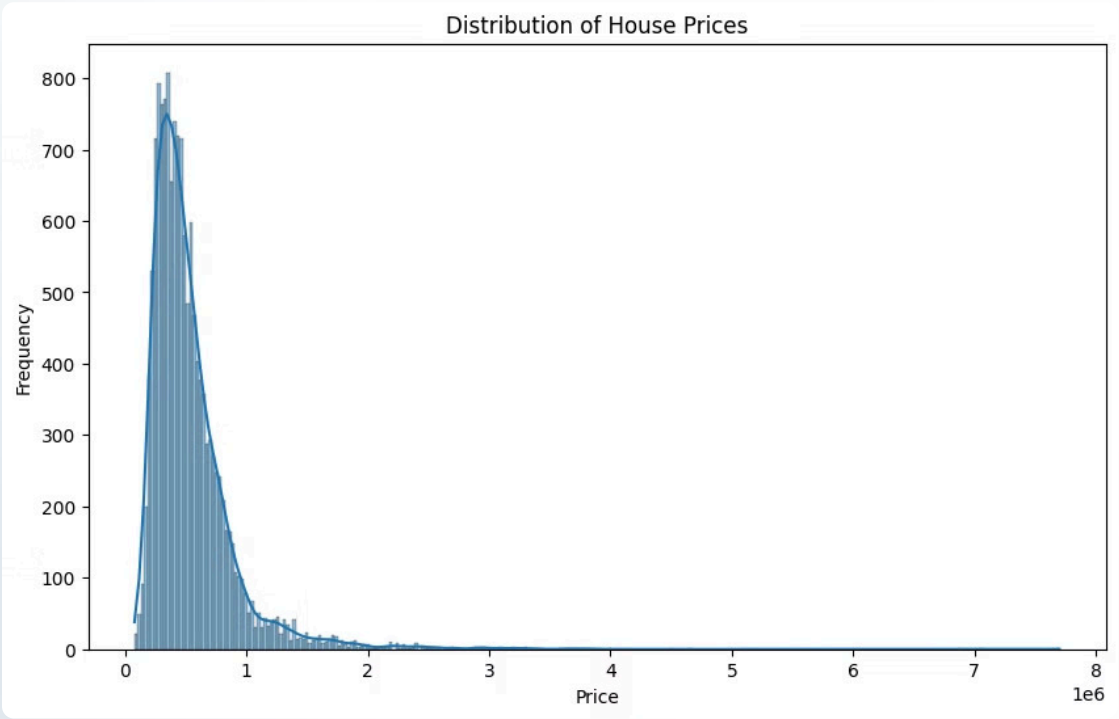
Data Visualization

1. House Price Distribution
2. Price vs. Living Area
3. Price vs. Number of Bedrooms
4. Outliers in House Prices

House Price Distribution

```
#histogram for distribution of House Prices
plt.figure(figsize=(10, 6))
sns.histplot(house['Price'], kde=True)
plt.title('Distribution of House Prices')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```

Here we found a positively skewed distribution, meaning the majority of houses are priced on the lower end (near the left of the x-axis), with fewer houses in the higher price ranges (towards the right).



Price vs Living Area

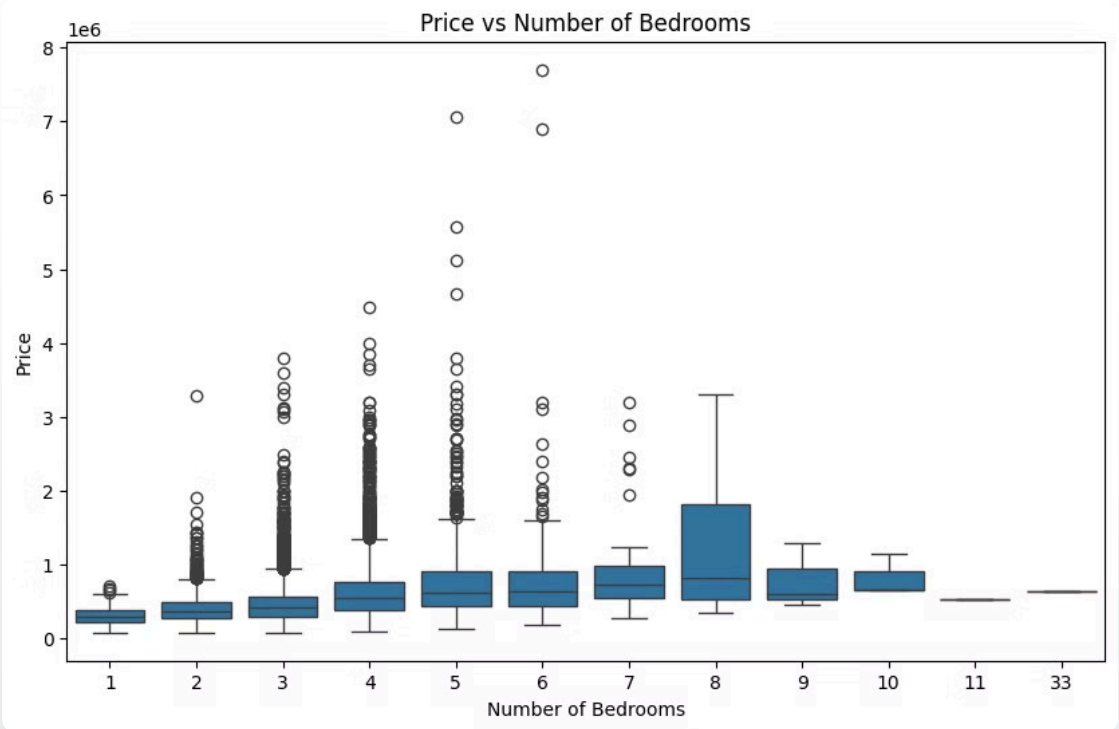
```
#scatterplot for Price vs Living Area
plt.figure(figsize=(10, 6))
sns.scatterplot(x=house['living area'], y=house['Price'])
plt.title('Price vs Living Area')
plt.xlabel('Living Area (sq ft)')
plt.ylabel('Price')
plt.show()
```

A scatterplot indicates a positive correlation between price and living area.

Price vs No. of bedrooms

```
# boxplot for price vs no. of bedrooms
plt.figure(figsize=(10, 6))
sns.boxplot(x=house['number of bedrooms'], y=house['Price'])
plt.title('Price vs Number of Bedrooms')
plt.xlabel('Number of Bedrooms')
plt.ylabel('Price')
plt.show()
```

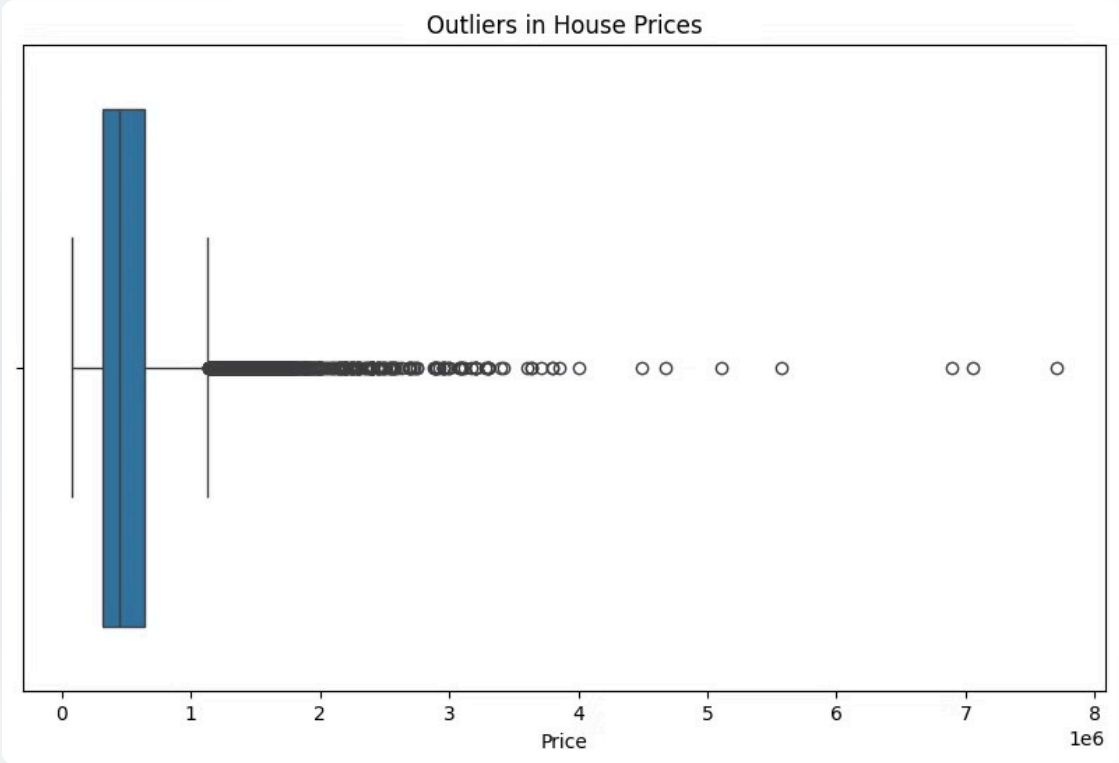
A boxplot shows how house prices vary by the number of bedrooms, with higher median prices for houses with more bedrooms.



Outliers in the House Price

```
#boxplot for outliers in house prices
plt.figure(figsize=(10, 6))
sns.boxplot(x=house['Price'])
plt.title('Outliers in House Prices')
plt.xlabel('Price')
plt.show()
```

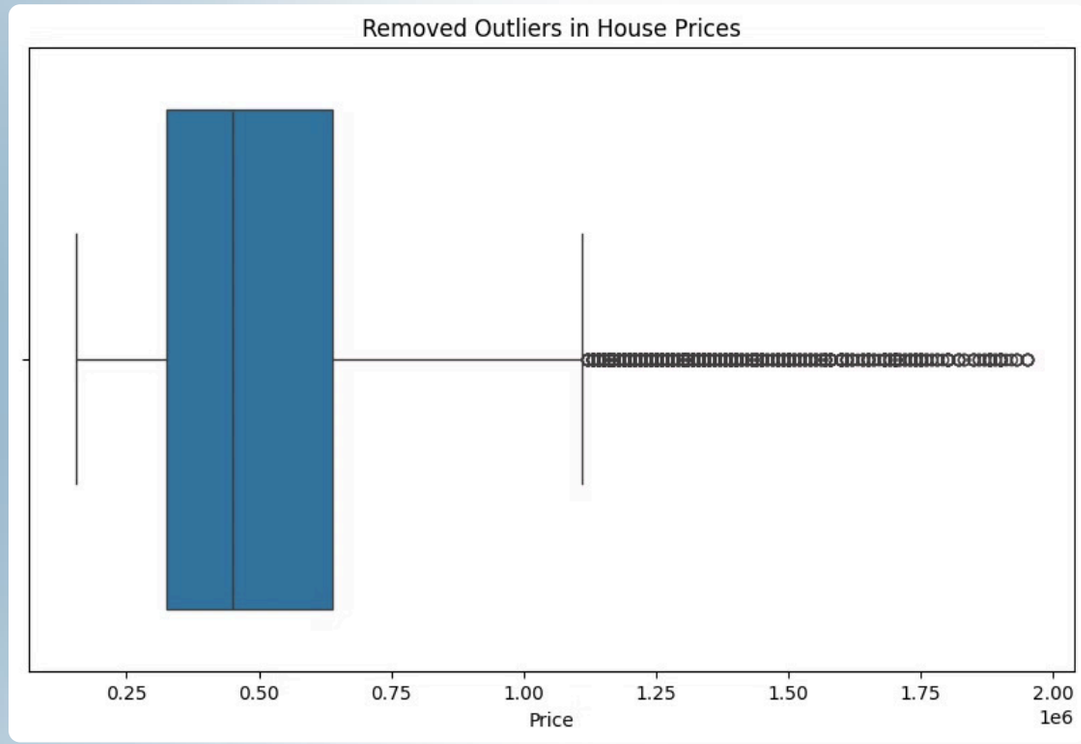
A boxplot highlights extreme values in price distribution, which could skew predictions if not handled.



Handling Outliers

```
[ ]  #handling outliers

     lower_bound = house['Price'].quantile(0.01)
     upper_bound = house['Price'].quantile(0.99)

     # Filter data based on percentile bounds
     no_outliers = house[(house['Price'] >= lower_bound) & (house['Price'] <= upper_bound)]

     # Display the updated dataset shape
     no_outliers.shape

     (14329, 23)
```

```
#boxplot for removing outliers in house prices
plt.figure(figsize=(10, 6))
sns.boxplot(x=no_outliers['Price'])
plt.title('Removed Outliers in House Prices')
plt.xlabel('Price')
plt.show()
```



Methodology

Outliers were identified and removed by defining upper and lower bounds based on percentiles.

Impact

The dataset shape was reduced to minimize model bias and improve prediction accuracy.

Data Preparation

Selected Features

Features like number of bedrooms, living area, house condition, and distance from airport, etc. were selected for modeling.

Splitting Data

The data was split into training and testing sets for accurate model evaluation.

```
#filtering out important columns
cols = ["number of bedrooms","number of bathrooms", "living area", "lot area", "number of floors", "condition of the house",
        "grade of the house","Area of the house(excluding basement)","Area of the basement", "Built Year",
        "Distance from the airport", "Number of schools nearby", "lot_area_renov","Price" ]
ndt = no_outliers[cols]
ndt.shape
```

```
(14329, 14)
```

```
from sklearn.model_selection import train_test_split
x = ndt.drop('Price', axis=1)
y = ndt['Price']

#splitting the data in train and tes 80% asnd 20%

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=21)
```

Models Used

1 Random Forest Regressor

An ensemble model combining multiple decision trees, handling non-linear relationships well.

2 Ridge Regression

Linear regression with L2 regularization, controlling overfitting.

3 Lasso Regression

Linear regression with L1 regularization, performing feature selection by shrinking less important variables to zero.

4 Linear Regression

A simple linear model assuming linear relationships, easy to interpret and quick to compute.

Lasso Regression

```
lsr = Lasso()

lsr.fit(x_train, y_train)
y_pred_lasso = lsr.predict(x_test)

r2_score_lasso = r2_score(y_test, y_pred_lasso)
acc_dict['Lasso'] = r2_score_lasso
r2_score_lasso
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/
  model = cd_fast.enet_coordinate_descent(
0.6440266661350058
```

Linear Regression

```
[27] lr = LinearRegression()

     lr.fit(x_train, y_train)
     lr_pred = lr.predict(x_test)

     lr_r2_score = r2_score(y_test, lr_pred)
     acc_dict['Linear Regression'] = lr_r2_score
     lr_r2_score
```

```
0.6440266969176394
```

Ridge Regression

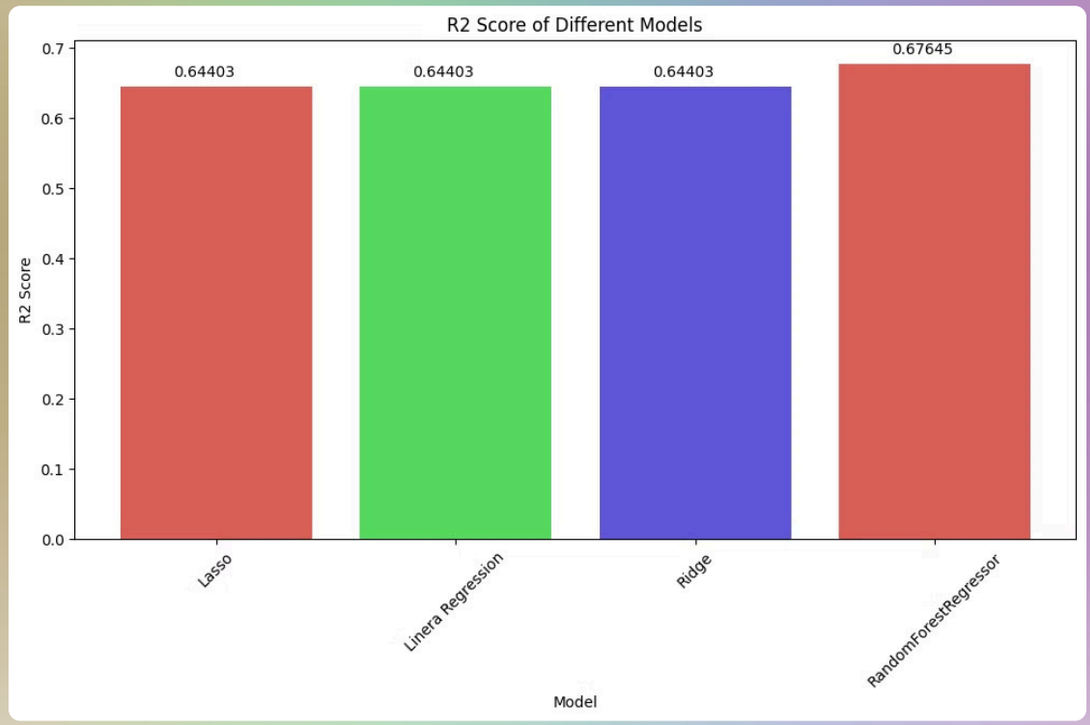
```
[ ]  rr = Ridge()

     rr.fit(x_train, y_train)
     ridge_pred = rr.predict(x_test)

     ridge_r2 = r2_score(y_test, ridge_pred)
     acc_dict['Ridge'] = ridge_r2
     ridge_r2
```

```
0.6440285022589874
```

Random Forest Regressor

```
[ ]  rfr = RandomForestRegressor()

     rfr.fit(x_train, y_train)
     rfr_pred = rfr.predict(x_test)
     rfr_r2 = r2_score(y_test, rfr_pred)
     acc_dict['RandomForestRegressor'] = rfr_r2
     rfr_r2
```

```
0.6748990246607938
```



Model Performance

0.644

Lasso

0.644

Linear

0.644

Ridge

0.674

Random Forest

Achieved the highest R^2 score, indicating superior performance in predicting house prices.

Cross-Validation

Why Cross-Validation?

Ensures model generalizes well to unseen data.

Results

Consistent performance across folds.

```
[ ] from sklearn.model_selection import cross_val_score
    cross_score = cross_val_score(estimator=rfr, X=x_train, y=y_train, cv=5)
    cross_score
```

```
array([0.67244363, 0.66511814, 0.66783564, 0.6435477 , 0.65658965])
```

```
[ ] cross_score.mean()
```

```
0.6611069504141313
```

Conclusion

1

Best Model

Random Forest Regressor emerged as the most accurate.

2

Applications

Assisting real estate professionals and homeowners in making informed decisions.

3

Future Work

Expanding the model's feature set and refining its accuracy.

Random Forest is the best-performing model for house price prediction.